

PowerPC 821/860 Diab Data Tools

0001003-001



Copyright (c) 1999
Accelerated Technology, Inc.
720 Oak Circle Dr. E.
Mobile, AL 36609
(334) 661-5770



Related Documentation

Nucleus PLUS Reference Manual, by Accelerated Technology, describes the operation and usage of the Nucleus PLUS kernel.

Nucleus PLUS Internals, by Accelerated Technology, describes, in considerable detail, the implementation of the Nucleus PLUS kernel.

Nucleus C++ Reference Manual, by Accelerated Technology, describes the operation and usage of the Nucleus C++ kernel.

Nucleus C++ Memory Management, Version 1.2, by Accelerated Technology, describes the operation and usage of the Nucleus C++ new and delete operators and C memory management.

Nucleus CLIB Reference Manual, by Accelerated Technology, describes the operation and usage of the Nucleus C -library.

Nucleus NET Reference Manual, by Accelerated Technology, describes the operation and usage of Nucleus NET.

Nucleus FILE Reference Manual, by Accelerated Technology, describes the operation and usage of Nucleus FILE.

Nucleus SMTP Reference Manual, by Accelerated Technology, describes the operation and usage of Nucleus SMTP.

Nucleus SNMP Reference Manual, by Accelerated Technology, describes the operation and usage of Nucleus SNMP.

Nucleus Telnet Reference Manual, by Accelerated Technology, describes the operation and usage of Nucleus Telnet.

Nucleus POP3 Reference Manual, by Accelerated Technology, describes the operation and usage of Nucleus POP3.

Nucleus PPP Reference Manual, by Accelerated Technology, describes the operation and usage of the Nucleus PPP Driver.

Nucleus FTP Server Reference Manual, by Accelerated Technology, describes the operation and usage of Nucleus FTP Server.

Nucleus FTP Client Reference Manual, by Accelerated Technology, describes the operation and usage of Nucleus FTP Client.

Nucleus TFTP Server Reference Manual, by Accelerated Technology, describes the operation and usage of Nucleus TFTP Server.

Nucleus TFTP Client Reference Manual, by Accelerated Technology, describes the operation and usage of Nucleus TFTP Client.

Nucleus WebServ Reference Manual, by Accelerated Technology, describes the operation and usage of Nucleus WebServ.



Style and Symbol Conventions

Program listings, program examples, filenames, menu items/buttons and interactive displays are each shown in a special font.

Program listings and program examples - `Courier New`

Filenames - `COURIER NEW, ALL CAPS`

Interactive Command Lines - **`Courier New, Bold`**

Menu Items/Buttons – *Times New Roman Italic*

Trademarks

MS-DOS is a trademark of Microsoft Corporation

UNIX is a trademark of X/Open

IBM PC is a trademark of International Business Machines, Inc.

SingleStep is a trademark of Software Development Systems.

Diab Data is a trademark of Diab Data, Inc.

Additional Assistance

For additional assistance, please contact us at the following:

Accelerated Technology

720 Oak Circle Drive, East

Mobile, AL 36609

800-468-6853

334-661-5770

334-661-5788 (fax)

support@atinucleus.com

<http://www.atinucleus.com>

Copyright (©) 1999, All Rights Reserved.

Document Part Number : 001003-001

Last Revised: October 25, 1999



Contents

Chapter 1- Introduction	1
Installing the Nucleus Software	2
Nucleus Distribution	2
Additional Documentation	3
Chapter 2 - Nucleus PLUS	5
Nucleus PLUS Software.....	6
Tools and Hardware	6
Building the Nucleus PLUS Library From a DOS Prompt.....	7
Command Line Parameters for Library Building	7
Conditional Compilation for Library Building.....	8
Building an Application	9
Command Line Parameters for Application Building.....	9
Conditional Compilation for Application Building	9
SingleStep Setup	10
Building and Executing the PLUS Demonstration System	
From a DOS Prompt	11
FADS board.....	11
EST board.....	11
MBX board.....	11
Running the Demonstration.....	12
RTA Integration.....	13
Profiling	13
Run-time Error Checking	15
Chapter 3 – Nucleus PLUS Usage.....	17
Data Types.....	17
System Startup.....	18
User Modifications.....	19
Execution Mode.....	19
Register Usage.....	19
Memory Usage	20
Nucleus PLUS Memory Usage.....	20



Stacks	20
Minimum Stack Size	22
Tuning Task/HISR Stack Size	22
Interrupt Vectors	22
Interrupt Control	25
Interrupt Control Services.....	25
Initial Interrupt Lockout	25
Timer Interrupt.....	26
Reentrancy	26
Chapter 4 – Nucleus PLUS Control Block Offsets	27
Task Control Block (TCB) Offsets.....	27
HISR Control Block (HCB) Offsets.....	29
Chapter 5 – Nucleus C++.....	31
Introduction	31
Tools and Hardware.....	32
Building the Nucleus C++ Library From a DOS Prompt.....	32
Compiler Options.....	33
Conditional Compilation	33
Building and Executing the C++ Demonstration System	34
System Startup	36
Operators New and Delete.....	37
Chapter 6 – Nucleus FILE.....	39
Introduction	39
Tools and Hardware.....	40
Building the Nucleus FILE Library From a DOS Prompt	41
Command Line Parameters.....	41
Conditional Compilation	41
Building and Executing the FILE Demonstration System From a DOS Prompt.....	42
Using the SingleStep Debugger	42
Architectural Considerations PCDISK.H file.....	43
Formatting A Disk.....	43
Device Driver Interface DEVTABLE.C.....	44
Initializing the Nucleus PLUS Environment NUFLC	44
Memory Allocation PC_MEMORY.C	44
The RAMDISK Demonstration Program	45
Interpreting RAMDEMO.C.....	45
RAMDEMO Initialization.....	45
Registering Tasks as File System Users	45
Input and Output for RAMDEMO.C.....	46
Restriction.....	46



Chapter 7 – Nucleus NET	47
Introduction	48
Tools and Hardware	49
Building The Nucleus NET Library From a DOS Prompt	49
Compilation Switches Used.....	49
Defines Used for Compilations	50
TARGET.H	51
Nucleus NET Interface	51
Overview of Demonstration System	51
Using the Nucleus NetDemo Application.....	51
Settings Common to all the NET Demos	53
TCP Server Demo	53
TCP Client Demo.....	57
UDP Client Demo	60
UDP Server Demo	62
Stopping the Test Application.....	67
Exiting the Program	67
Chapter 8 –Nucleus PQUICC Driver.....	69
Introduction.....	69
Tools and Hardware	70
Building The Nucleus PQUICC Library From a DOS Prompt.....	71
Description of Driver Files	71
Building and Executing the PQUICC Driver Demos From a DOS Prompt	72
Other Nucleus NET Interface Calls	73
Chapter 9 - Nucleus FEC860T Driver	75
Introduction.....	75
Tools and Hardware	76
Building The Nucleus FEC 860T Library From a DOS Prompt.....	77
Description of Driver Files	77
Building and Executing the FEC860T Driver Demos From the DOS Prompt	78
Other Nucleus NET Interface Calls	80
Chapter 10 – Nucleus PPP Driver	81
Introduction.....	81
Tools and Hardware	83
Building the Nucleus PPP Library From a DOS Prompt.....	83
Building and Executing the PPP Demonstration System From a DOS Prompt.....	84
Instructions for Setting Up TCPserv and UDPserv Demos in \NMODEM.....	84
Instructions for Setting Up TCPcli and UDPcli Demos in \DIALUP.....	86
Chapter 11 – Nucleus FTP Server	89
Introduction.....	89
Tools and Hardware	90
Building the Nucleus FTP Server Library From a DOS Prompt.....	91
Building and Executing the FTP Server Demonstration System From a DOS Prompt.....	91



Chapter 12 – Nucleus FTP Client.....	93
Introduction	94
Tools and Hardware	94
Building the Nucleus FTP Client Library From a DOS Prompt.....	95
Building and Executing the Nucleus FTP Client Demonstration System From a DOS Prompt.....	95
Chapter 13 – Nucleus TFTP Server.....	97
Introduction	98
Tools and Hardware	98
Building the Nucleus TFTP Library From a DOS Prompt	99
Building and Executing the TFTP Demonstration System From a DOS Prompt.....	99
Building the Demo	99
Executing the Demo	99
Chapter 14 – Nucleus TFTP Client.....	101
Introduction	102
Tools and Hardware	102
Building the Nucleus TFTP Library From a DOS Prompt	103
Building and Executing the TFTPDEMO Application From a DOS Prompt.....	103
Chapter 15 – Nucleus SNMP	105
Introduction	105
Tools and Hardware	106
Rebuilding Other Nucleus Libraries.....	107
Building the SNMP Library From a DOS Prompt	107
Building and Executing the SNMP Demonstration System From a DOS Prompt.....	107
Chapter 16 – Nucleus SMTP.....	109
Introduction	110
Tools and Hardware	110
Building the Nucleus SMTP Library From a DOS Prompt	111
Building and Executing the Nucleus SMTP Client Demonstration From a DOS Prompt.....	111
Building and Executing the Nucleus SMTP Server Demonstration From a DOS Prompt.....	112



Chapter 17 – Nucleus POP3	115
Introduction	115
Tools and Hardware	116
Building the Nucleus POP3 Library From a DOS Prompt	116
Building and Executing the POP3 Demonstration System	
From a DOS Prompt	117
Chapter 18 - Nucleus Telnet.....	119
Introduction	120
Tools and Hardware	120
Building the Nucleus Telnet Library From a DOS Prompt	121
Building and Executing the Telnet Demonstration System	
From a DOS Prompt	121
Chapter 19 - Nucleus RIP2.....	123
Introduction	123
Tools and Hardware	124
Building the Nucleus RIP2 Library From a DOS Prompt	125
Building and Executing the RIP2 Demonstration System	
From a DOS Prompt	125
Chapter 20 – Nucleus Webserv	127
Introduction	127
Tools and Hardware	129
Building the Nucleus WebServ Library From a DOS Prompt	129
Building and Executing the WebServ Demonstration System	
From a DOS Prompt	129
IN_MEMORY_FILE_SYSTEM.....	130
IF_NOT_USING_IN_MEMORY_FILE_SYSTEM (with Nucleus FILE)	130
Miscellaneous Information.....	132
Chapter 21 – Nucleus CLIB	133
Introduction	133
Tools and Hardware	134
Building the Nucleus CLIB Library From a DOS Prompt	135
Building and Executing the CLIB Demonstration System	
From a DOS Prompt	135
Chapter 22 - Nucleus EDE	137
Configuration of EDE	138
Demonstration System	138
Setting the Tool Directory	139
Exclude From Build.....	140
Starting a Debugger through EDE	141





1

Introduction

Installing the Software

Nucleus Distribution

Additional Documentation



Installing the Nucleus Software

The Nucleus distribution software for the Power PC821/860 environment contains all the files needed for operation, as well as an example system for each of the products you have purchased. You will need to run the program "SETUP.EXE," which is located on the CD you received from Accelerated Technology, Inc. After running this program, follow the directions on your screen. This program will install the software you have purchased and place it in the directory you have provided.

Nucleus Distribution

Nucleus PLUS and ATI's other products expect a particular directory structure to exist for the build process to operate correctly. You are free to modify this directory structure, however, some modification will be necessary for a clean operation of the associated batch files and/or project files.

The directory structure written by ATI's installation program will be as follows:



This is only a sample listing of ATI's products. The actual directory structure may be different, depending on the products you have purchased.



Additional Documentation

Additional documentation for any products that you have purchased can be located in the Docs directory where the Nucleus software was installed. As an example, this directory defaults to C:\ATI. Therefore, additional documentation would be located in the C:\ATI\DOCS directory.

NOTE: Command line instructions for building libraries and demos do not apply to Nucleus EDE ports. Refer to your Nucleus EDE On-Line Help and Chapter 22 for more information.





Nucleus PLUS

Nucleus PLUS Software

**Building the Nucleus PLUS
Library From a DOS Prompt**

Building an Application

**Building and Executing the
Demonstration System From a
DOS Prompt**

Running the Demonstration

RTA Integration

Profiling

Run-time Error Checking



Nucleus PLUS Software

The Nucleus PLUS distribution software for the PowerPC 821/860 environment contains all the files needed for operation, as well as an example system. Information in this chapter deals with specific requirements of the PowerPC 821/860 using Diab Data tools. Examples in the following sections are given for Diab Data tools hosted on a Windows system.

NOTE: Instructions for building the library and the application do not apply to Nucleus EDE ports. Please refer to your Nucleus EDE On-Line Help and Chapter 22 for more information.

To install the Nucleus PLUS software, simply execute the `SETUP.EXE` from the distribution software. The Nucleus PLUS library and demo rely on this structure and will not build correctly if this directory structure is modified. After installation, the directory structure should resemble the following:



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.3B
Assembler:	DAS V4.3B
Linker:	DLD V4.3B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST and MBX boards are compile-only.)



Building the Nucleus PLUS Library From a DOS Prompt

Nucleus PLUS is implemented as a C library. The MS-DOS batch file `PLUS.BAT` contains the Diab Data assembler, compiler, and archiver commands necessary to build the Nucleus PLUS archive.

`PLUS.BAT` creates the Nucleus PLUS library (`\PLUS\O\PLUS.LIB`). The default compiles for the MPC860 FADS board. If you have one of the other 821/860 boards use one of the following as a command-line parameters:

FADS - Motorola 821/860 FADS board

EST - Motorola 821/860 EST board

MBX - Motorola 821/860 MBX board

The Nucleus library can be built by executing the following command :

```
\ATI\PLUS> plus fads <cr> -- Builds for the Motorola FADS board.
```

or

```
\ATI\PLUS> plus est <cr> -- Builds for the EST SBC860 board.
```

or

```
\ATI\PLUS> plus mbx <cr> -- Builds for the Motorola MBX board.
```

The previous commands assume the existence of two files, namely `DEMO.LNK` and `DEMO.C`

The file `CRT0.S` is supplied by the Diab Data tools and provides some run-time initialization. `INT_FADS.S`, `INT_EST.S` or `INT_MBX.S`, contains all of the low-level Nucleus initialization and the hardware initialization for the evaluation board. `USER_MMU.C` and `MMU.S` contain the initialization of the memory management unit and the setup of user-defined memory regions in the translation lookaside buffers. `UART.C` provides a driver for the SMC1 and SMC2 channels on the MPC860.

Command Line Parameters for Library Building

There are several command line parameters associated with compilation / assembly / archiving of each library element. The following is a description of each command line option for the Diab Data development tools on an MS-DOS host:



Command Line Option	Meaning
-c	Specifies an ".o" file, bypassing the link phase of the compiler.
-g	Generates symbolic debugger information, with most optimization turned off.
-r	Replaces the named ".o" files in the archive "PLUS.LIB"
-sR	Updates the symbol table in the archive file and sorts the object files in the archive.
-tPPC860ES	Compiler generates an ELF format object file and uses software floating-point.
-DFADS	Used in USER_MMU.C and UART.C for FADS board.
-DEST	Used in USER_MMU.C and UART.C for EST board.
-DMBX	Used in USER_MMU.C and UART.C for MBX board.

Conditional Compilation for Library Building

The Nucleus PLUS library has several conditional compilation options. These conditional compilation flags enable various features within the Nucleus PLUS library source.

The conditional compilation options and their specification with the Diab Data development tools is as follows:

Compilation Flag	Meaning
-DNU_ENABLE_HISTORY	Enables history saving in the specified file. Note: only files of the form *.C are affected by this option.
-DNU_ENABLE_STACK_CHECK	Enables stack checking at the beginning of each function in the specified file. Note: only files of the form *.C are affected by this option.
-DNU_ERROR_STRING	Enables making an ASCII error string if a fatal system error occurs. This flag is applicable to the compilation of ERD.C, ERI.C, and ERC.C.
-DNU_NO_ERROR_CHECKING	Disables error-checking shell on creation of the timer HISR in TMI.C. Not applicable to any other Nucleus PLUS library compilation.
-DNU_DEBUG	Maps the data structures used by the application and defined in NUCLEUS.H to the actual internal data structures in Nucleus PLUS. If this option is used, it is typically a good idea to compile the entire library with it.
-DNU_INLINE	Replaces some linked-list processing with in-line code in order to improve performance. This option is applicable to any or all *.C or *.S files.



Building an Application

Building an application is described in the “Getting Started” chapter of the *Nucleus PLUS Reference* manual. Assuming that `NUCLEUS.H`, `PLUS.LIB` and `DEMO.C` are accessible from the current directory, the following Diab Data commands (in **bold type**) would build and link a simple user application:

```
> dcc -v -g -tPPC860ES -c -X23=0 demo.c
> dld -lc -m2 -o demo.elf demo.o plus.lib demo.lnk > o\demo.map
```

The `DEMO.LNK`, file is used by the linker to handle all the loading of files, linking any of the routines needed from the archive, supplying starting addresses for memory, variables, the `.bss` section, etc. The file `DEMO.C` contains the function, `Application_Initialize`, as well as all tasks and other structures. The `DEMO.O` file needs to be linked together, along with `PLUS.LIB`.

Command Line Parameters for Application Building

There are several command line parameters associated with compilation/assembly of application elements. The following is a description of each command line option for the Diab Data development tools on an MS-DOS host:

Command Line Option	Meaning
-c	Specifies an “.O” file, bypassing the link phase of the compiler.
-g	Generates symbolic debugger information, with most optimization turned off.
-o	Specifies the output file name for the linker.
-m2	Generates a map file.
-tPPC860ES	Compiler generates an ELF format object file and uses software floating-point.

Conditional Compilation for Application Building

Nucleus PLUS application elements may disable error checking on parameters supplied to Nucleus PLUS services by supplying the **-DNU_NO_ERROR_CHECKING** command line option to the compiler. This results in a substantial increase in run-time performance, and also may reduce code size.

The application data structures defined in `NUCLEUS.H` may be mapped directly to the internal Nucleus PLUS data structures by defining `NU_DEBUG` with a **-D** option during compilation. This allows the user to directly examine the internals of each Nucleus PLUS data structure within a source-level debugging environment. If the `NU_DEBUG` option is used, it is often a good idea to re-build all of the Nucleus PLUS source code with the `NU_DEBUG` option.



SingleStep Setup

Prior to launching the SDS 7.4 debugger, SDS must be set up to properly configure the board which you will be using using a configuration file. Go to the \INIT subdirectory of SingleStep and create or modify SSTEP.INI. SSTEP.INI will contain one line that consists of the following:

```
alias _config 'source {dir}\cmd\{board}.cfg'
```

{dir} is the directory where Singlestep is installed. {board} is the configuration filename. Look in the 'CMD' directory for the file that corresponds to your particular board.

Depending on the version of the SingleStep debugger, they should be similar to the following:

FADSRZ0220.CFG	for the Motorola 821/860 FADS board
ESTSBC8X.CFG	for the Motorola 821/860 EST board
MBX860.CFG	for the Motorola 821/860 MBX board

SDS 7.4 may also require a second file that needs to be present in the \CMD directory along with the .CFG file. Depending on the version of the SingleStep debugger and which evaluation board is being used, the file will be similar to one of the following:

UPMFADS0220.DBG	for the Motorola 821/860 FADS board
ESTUPM.DBG	for the Motorola 821/860 EST board

All of the files can be found on the distribution CD you received from ATI, in the folder \ATI\SDS, or on the internet at <http://www.sdsi.com>. There may be additional .CFG or .DBG files. You need the .CFG and .DBG files that match your board.

Note: When SingleStep starts up and takes control of the board, it initializes the DER register to a value that causes all interrupts to be trapped by the debugger. The Nucleus demo requires the decremter interrupt for correct operation, so the initialization value for the DER must be changed to allow the decremter interrupt to be handled by Nucleus. This should be done by modifying the configuration file that SingleStep uses to initialize the hardware. The configuration file has the extension .CFG and is normally found in the \cmd subdirectory of the SingleStep product installation. In the configuration file, the following line

```
@ DER = 0xFFE7400F
```

should be changed to

```
@ DER = 0xFFC7400F
```

After this change has been made, SingleStep will be able to initialize the hardware without trapping the decremter or external interrupts.



Building and Executing the PLUS Demonstration System From a DOS Prompt

The Nucleus PLUS software contains all the files necessary to build a demonstration system. The demonstration system is designed to execute on the Motorola FADS board, the EST SBC860 board, or the Motorola MBX board using the SingleStep debugger.

The Nucleus demonstration system can be built by executing the following command:

```
\ATI\PLUS\DEMO> demo <cr>
```

The DEMO.BAT batch file compiles the DEMO.C file. The demo batch files then links all the files together and generates the map file and the ELF file for downloading to the board. This build process should be used as an example for any user-created applications.

An absolute file, named DEMO.ELF, is produced by the DEMO.BAT batch file. This file is ready to be downloaded by the SingleStep On Chip (MPC8XX) BDM debugger. Once the SingleStep BDM has been started, it will prompt you for the demo file you wish to load or you may load the file by doing the following:

- Choose *File* and then select *Debug* in the SDS debugger to upload the O\DEMO.ELF executable to the target.
- Select the *Processor* tab, and select “By Object File”.
- Select the *Options* tab and uncheck “Require Exact Symbol Names”.
- The remaining options are checked. Click *OK* to upload.

After the code has downloaded, click on the “green light” button to start the demo. The demo prints status information out of the serial port. Depending on what board is being used, the serial demo may be viewed by one of the following ways:

FADS board

Connect a null modem cable from the bottom DB9 port of the FADS board to an available COM port of your PC.

EST board

Connect a null modem cable from the JP3 port of the EST board to an available COM port of your PC.

MBX board

Connect a null modem cable from the SMC1 port of the MBX board to an available COM port of your PC.



Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

Then use a terminal program (e.g., Hyperterm in Windows 95) at 115200-8-N-1 to connect to the COM you are using. Attach a null modem adapter to the cable if there is no output.

Observe the text displayed in the terminal window. This text will print at approximately one-second intervals. Below is an example of the text that is displayed.

```
*****
Copyright (c) 1993-1999 ATI - Nucleus PLUS - Version MPC860/D 1.12.7
*****

System Variable Status:
Task 0 time:                1
Task 1 messages sent:       15648
Task 2 messages received:   15605
Task 2 invalid messages:    0
Who has the resource:       Task 3
Event detections:           0
Press Q to exit!
```

Running the Demonstration

After the demo program has run for a few minutes, there are some variables that can be examined to ensure the program is running properly. These variables should be put into the *Watch Window* of the debugger. The variables are summarized in the following table.

Variable	Function	Description
Task_Time	task_0	Number of times task_0 has been to sleep and awakened.
Task_1_messages_sent	task_1	Number of messages task_1 has sent to task_2.
Task_2_messages_received	task_2	Number of messages task_2 has received from task_1.
Task_2_invalid_messages	task_2	Number of unsynchronized messages task_2 has received from task_1.
Event_Detections	task_5	Number of times task_5 has received an event from task_0.

Task_Time and Event_Detections should be equal, or differ by no more than one.
Task_2_messages_received should be within 100 messages of Task_1_messages_sent.
Task_2_invalid_messages should be 0.



RTA Integration

This port supports the profiling and run-time error checking features of the Diab Data RTA suite. In order to use these features, you must have the following:

- Diab Data 4.3B or higher
- SingleStep 7.4 or higher
- You must have installed the RTA component of SingleStep
- If you are using Nucleus EDE, it must be 3.1 or higher

By default, Nucleus PLUS will not have the RTA features enabled when it is compiled as described on the preceding pages. The RTA features are enabled by compiling the PLUS library and application with additional flags. Batch files are provided which compile the PLUS library and application with the added flags along with the default flags. Nucleus PLUS should be enabled with only one of the RTA features at a time, NEVER both of them at the same time.

Profiling

To instrument all of the generic Nucleus PLUS code for profiling, use `PLUSPROF.BAT` in place of `PLUS.BAT`. `PLUSPROF.BAT` is used exactly like `PLUS.BAT` as described in the section on building the Nucleus PLUS library. To build the demonstration program for profiling, use `DEMOPROF.BAT` in place of `DEMO.BAT`. When building any application, `DEMOPROF.BAT` should be used as a reference. Any C files that you want profiled must be built with the `-Xprof-all` and `-DNU_RTA_PROF` flags in addition to the default flags.

The application modules must be linked with the RTA library in addition to the PLUS library, by adding the `-lrtc` flag to the linker command.

There are two utilities required by RTA for it to gather profiling data from the target and display the data in graphical form. These programs are provided in the `\PLUS` directory of the Nucleus PLUS distribution.

`RTASSTEP.EXE` should be copied to the `\DIAB\4.3B\WIN32\TARGCOMM\SSTEP` directory. `PROFILER_COLLECT_DATA.DBG` should be copied to the `\CMD` directory of the SingleStep installation. These utilities may be modified and updated by Diab Data, in which case you should use the Diab versions instead of the ones provided in the Nucleus PLUS distribution.

If you are using Windows NT, you must make the `LM_LICENSE_FILE` environment variable that Diab Data uses available to the NT system. From the *Start* menu, choose *Settings/Control Panel*, then double-click the *System* icon. In the resulting dialogue box,



Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

choose the *Environment* tab, and add the `LM_LICENSE_FILE` environment variable, setting it to point to the Diab Data license file.

If you are using Windows 95/98, make sure `LM_LICENSE_FILE` is in `AUTOEXEC.BAT`. The demonstration program `DEMO.ELF` should be downloaded to the target board as described in the preceding pages. Make sure to enable the *Require Exact Symbol Names* option under the *Options* tab of the *Debug* dialog box. After the program has been downloaded and the *Debug* dialogue box has been cleared from the screen, the RTA component of SingleStep must be configured. In the button bar, SingleStep provides three RTA-related buttons. If these buttons do not appear, choose RTA PROFILER in the *TOOLBARS* menu to enable them.

Click on the first RTA button to bring up the RTA configuration dialogue box. Assuming you are using the default installation directories, enter the following values:

Profiler Path: `C:\DIAB\4.3B\WIN32\BIN\RTA.EXE`

Command Parameters: `-cd c:\ati\plus\demo\o`
`c:\ati\plus\demo\o\rtadb`

Profiler data file: `C:\ATI\PLUS\DEMO\O\DEMO.PRF`

Although these values are for the Nucleus PLUS demo, they should be referred to as an example for your own application.

After the RTA has been configured, the program you downloaded can be started. While it runs, profiling data will be saved on the target. After stopping execution, bring up the SingleStep console window, then click on the third RTA button. The debugger will print a message to the console, upload the profiling data from the target, then print a message indicating that the data has been uploaded.

Click on the second RTA button to start the RTA graphical program display. If the RTA program is running for the first time, it will ask to create a directory to store user information. If RTA doesn't find a previously created `\RTA_DB` subdirectory, it will ask to create one as specified by the *Command Parameters* setting of the RTA configuration.

Choose *File/Open* from the menu and load `DEMO.ELF`. Choose *Application/Target Settings...* from the menu. In the resulting dialogue box, highlight `sds_singlestep`, then click on the *OK* button. You are now ready to connect to an RTA server. Choose *Application/Connect* from the menu to start the RTA server. The RTA server will print messages to the RTA output log window, the last one indicating it is ready for commands. To read and display the profiling data, choose *Application/Snapshot* from the menu. Please refer to the RTA user's manual for details about the graphical displays and other features of RTA.



Run-time Error Checking

To instrument all of the generic Nucleus PLUS code for run-time error checking, use `PLUSRTC.BAT` in place of `PLUS.BAT`. `PLUSRTC.BAT` is used exactly like `PLUS.BAT` as described in the section on building the Nucleus PLUS library.

By default, `DEMO.C` does not contain any code that the run-time error checker will complain about. To see the run-time error checker actually flag a problem, insert the following line of code into any one of the functions of `DEMO.C`:

```
NU_Deallocate_Memory(&Task_Time);
```

This line of code attempts to free global memory, which is a violation the run-time error checker will report.

To build the demonstration program for run-time error checking, use `DEMORTC.BAT` in place of `DEMO.BAT`. When building any application, `DEMORTC.BAT` should be used as a reference. Any C files that you want checked at run-time must be built with the `-xrtc` and `-DNU_RTA_RTC` flags in addition to the default flags. The application modules must be linked with the RTA library in addition to the PLUS library, by adding the `-lrta` flag to the linker command.

The demonstration program `DEMO.ELF` should be downloaded to the target board as described in the preceding pages. Make sure to enable the *Require Exact Symbol Names* under the *Options* tab of the *Debug* dialogue box. After the program has been started, bring up the SingleStep console window to see the message the RTA printed about the memory violation. All of the run-time error messages are displayed on the SingleStep console. Please refer to the RTA user's manual for more information on the types of errors the run-time error checker deals with and the messages it displays.





Nucleus PLUS Usage

Data Types

System Startup

Execution Mode

Register Usage

Memory Usage

Nucleus PLUS Memory Usage

Stacks

Interrupt Vectors

Interrupt Control

Interrupt Handling

Timer Interrupt

Reentrancy

Data Types

Since different processor architectures and even different compilers define C data types differently, Nucleus PLUS defines a set of its own data types. These data types are mapped to the compiler data types that have the required capability.

Nucleus PLUS defines several data types in the include file `NUCLEUS.H`. The data type assignments for the Diab Data compiler are as follows:

Nucleus PLUS Type	Actual Data Type
UNSIGNED	unsigned long
SIGNED	long
OPTION	unsigned long
DATA_ELEMENT	unsigned long
STATUS	int
UNSIGNED_CHAR	unsigned char
CHAR	char
INT	int
VOID	void
UNSIGNED_PTR	unsigned long *
BYTE_PTR	unsigned char *
INT8	char
UINT8	unsigned char
INT16	short int
INT32	unsigned short
INT32	long int
UINT32	unsigned long

System Startup

The label `start` is defined at the reset vector and marks the entry point of the system. The reset vector calls the Nucleus function `INT_Initialize`.

`INT_Initialize` is responsible for initializing the board, setting up the interrupt vector table, and setting up several system stack areas. `INT_Initialize` calls the Diab Data function `_init_main`, which copies the data section from ROM to RAM and clears the `.bss` section. After `_init_main` returns to `INT_Initialize`, control is transferred to `INC_Initialize` for complete, high-level initialization of the Nucleus PLUS system. `INC_Initialize` calls the `Application_Initialize` function after all other initialization is complete, and before the Nucleus PLUS scheduler is invoked.



User Modifications

Configuring Nucleus PLUS and the PowerPC startup routines to a new target environment might require slight source modifications. Please see the Diab Data PowerPC documentation/source for a related discussion. The following files might require minor modifications.

File	Meaning
INT_FADS.S, INT_EST.S, or INT_MBX.S	This is the Nucleus PLUS initialization file. Typically, this file contains most low-level initialization.
USER_MMU.C	This is the MMU initialization file. It contains the table of user-defined memory regions.

Execution Mode

All execution in the Nucleus PLUS system is done with the MSR of the PowerPC set to 0x1032. This value enables the Machine Check state, enables the MMU, sets non-maskable interrupts to a recoverable state, and bases the vector table at address 0x0. When interrupts are enabled, the MSR is set to 0x9032 to allow External Exception interrupts. This includes initialization, task execution, and interrupt processing.

Register Usage

Nucleus PLUS uses general purpose registers as outlined in the following table.

Register	Usage
r0	Scratch register. Not preserved across function boundaries.
r1	Stack Pointer.
r2	Reserved
r3 - r12	Temporary registers. Not preserved by functions, or across function boundaries. Holds variables whenever possible.
r3 - r10	Used for parameter and result passing.
r13	Reserved.
r14 - r31	Preserved registers. Saved when used by functions. Contains variables that cannot be placed in r3 - r12.



Memory Usage

Nucleus PLUS, like any program, requires memory for instructions and data. Memory for instructions, global data, and static data is allocated during the link phase of building a system. During system startup, Nucleus PLUS performs a minimal amount of run-time memory allocation. All subsequent memory allocation is under the control of the application.

The linker decides where to locate memory SECTIONS in a given target environment. A memory SECTION is basically a collection of similar data elements grouped together by the linker, i.e. data, text, and bss. For example, all processor instructions are grouped together in the `".text"` area. All initialized data is in the `".data"` area, and the uninitialized data is in the `".bss"` area.

Nucleus PLUS Memory Usage

Nucleus PLUS performs some minimal memory allocation after the end of the `__SP_INIT` area. This area is the starting point of the system stack and is placed just after the `".text"` and `".bss"` sections of memory. The starting addresses and sizes of each section are set up in the `DEMO.LNK` file. Nucleus PLUS allocates the following areas, which are each defined in `INT_FADS.S`, `INT_MBX.S` or `INT_EST.S` :

- System stack area
- IRQ stack area
- Timer HISR stack area

The address after the previously mentioned allocation is passed to `Application_Initialize` as the first available memory location.

Stacks

There are principally three types of stacks in a Nucleus PLUS system, namely the system, task, and High-Level Interrupt Service Routine (HISR) stacks. The system stack is defined and allocated in `INT_FADS.S`, `INT_MBX.S` or `INT_EST.S`. Task and HISR stacks are determined by the user during creation.

The system stack is used during initialization and while executing in the scheduling loop. Additionally, the system stack is used during execution of Low-Level Interrupt Service Routines (LISRs). The size of the system stack should take into account the worst case interrupt requirements in the system, including nested interrupt conditions.



The context of each task or HISR is saved on its respective stack. If a task or HISR is interrupted, all registers are saved on the stack. Otherwise, if a task suspends or relinquishes control as a result of a Nucleus PLUS service call, only the registers that must be preserved across function boundaries are saved. The size of task and HISR stacks must be large enough to accommodate all local variable allocation and maximum function call nesting, in addition to the space necessary to save all of the PowerPC 821/860 registers.

The first word of the stack of a non-executing task or HISR determines the type of stack format present. If the first word contains a value of one, an interrupt stack frame is present. Otherwise, if the first word of the stack is zero, a non-interrupt stack frame is present.

Interrupt Stack Frame:

tc_stack_pointer	->	Next available space
		DIAB/DATA old SP if being saved
		DIAB/DATA old LR which is saved
		Stack type (Interrupt Stack Type 1)
		SRR1/MSR
		Registers r0,r3-r8,r14-r31
		CTR
		XER
		CR
		LR
		SRR1/MSR
		SRR0/TCC_Task_Shell
		r9
		r10
		r11
		r12
		DIAB/DATA old SP if being saved
		DIAB/DATA old LR which is saved

Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

Solicited Stack Frame:

tc_stack_pointer	->	Next available space
		DIAB/DATA old SP if being saved
		DIAB/DATA old LR which is saved
		Stack type (Interrupt Stack Type 0)
		MSR
		Registers r14-r31
		CR
		XER
		LR
		DIAB/DATA old SP if being saved
		DIAB/DATA old LR which is saved

Minimum Stack Size

Nucleus PLUS is configured to not allow task or HISR stacks that are less than 172 bytes in size. Additionally, a stack overflow condition is considered present if there are fewer than 164 bytes remaining on the non-floating point task or HISR stack.

Tuning Task/HISR Stack Size

Internal stack checking and the service `NU_Check_Stack` check for stack overflow conditions. At the same time, the minimum amount of available stack space is monitored. This value can be used to determine how much stack space is needed by a given task or HISR.

Interrupt Vectors

Nucleus PLUS interrupt vectors are defined in `INT_XXX.S` at the label `INT_Vectors`. The entire Nucleus vector table is located at address 0x0 during the linking stage. During the run-time Nucleus initialization, the IP bit of the MSR register is cleared to force all exceptions to vector from base address 0x0.

LISRs can be registered for all interrupts except system reset (0x100), and decrementer (0x900). The system reset vector jumps to `INT_Initialize`. Nucleus uses the decrementer for its periodic timer interrupt.



The following is a list of all the MPC860 interrupt sources and their corresponding Nucleus vector numbers:

Interrupts	Nucleus Vector Number
Core	
Machine check	2
Alignment	6
Program error	7
System call	12
Trace	13
Software emulation	16
Instruction TLB miss	17
Data TLB miss	18
Instruction TLB error	19
Data TLB error	20
Data breakpoint	28
Instruction breakpoint	29
Peripheral breakpoint	30
Non-maskable dev port	31

Interrupts	Nucleus Vector Number
SIU	
IRQ 0	32
Level 0	33
IRQ 1	34
Level 1	35
IRQ 2	36
Level 2	37
IRQ 3	38
Level 3	39
IRQ 4	40
Level 4	41
IRQ 5	42
Level 5	43
IRQ 6	44
Level 6	45
IRQ 7	46
Level 7	47



Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

Interrupts	Nucleus Vector Number
CPM	
Error	64
PC4	65
PC5	66
SMC2/PIP	67
SMC1	68
SPI	69
PC6	70
Timer 4	71
PC7	73
PC8	74
PC9	75
Timer 3	76
PC10	78
PC11	79
CPM	
I2C	80
RISC timer table	81
Timer 2	82
IDMA2	84
IDMA1	85
SDMA Channel Bus error	86
PC12	87
PC13	88
Timer 1	89
PC14	90
SCC4	91
SCC3	92
SCC2	93
SCC1	94
PC15	95



Interrupt Control

Interrupts may be disabled and enabled, for brief periods, by Nucleus PLUS. The interrupts disabled in a PowerPC 860 environment are defined by `NU_DISABLE_INTERRUPTS` and `NU_ENABLE_INTERRUPTS`. By default, these are set to 0x0 and 0x8000 respectively.

Interrupt Control Services

Nucleus PLUS allows the user to enable and disable PowerPC 860 interrupts. The values passed to the interrupt control routines represent the interrupt lockout bits to be placed in the PowerPC 860 Machine State Register (MSR). Furthermore, the interrupt lockout bits must be in the same location as they are in the MSR.

There are two default parameters to the interrupt control functions (`NU_Control_Interrupts` and `NU_Local_Control_Interrupts`), as follows:

Default Parameter	Value	Meaning
<code>NU_ENABLE_INTERRUPTS</code>	0x8000	Enables External
<code>NU_DISABLE_INTERRUPTS</code>	0x0000	Disables External

Interrupt control services may be called from application tasks, LISRs, and HISRs. Interrupt control services are not allowed from the `Application_Initialize` function. It is important to realize that interrupts enabled or disabled by `NU_Control_Interrupts` remain enabled or disabled until a subsequent call is made. There is one exception to this rule. If this function is called from an LISR in a nested interrupt condition, the previously interrupted ISR's interrupt level is restored after the current LISR finishes.

Initial Interrupt Lockout

After initialization is complete, Nucleus PLUS places the value of `NU_ENABLE_INTERRUPTS` into the MSR. Hence, before any task or HISR executes, interrupts are enabled to this level.



Timer Interrupt

Nucleus PLUS requires a periodic interrupt in order to provide time-oriented services such as time-slicing, service call timeouts, and application timers. Initialization and interrupt vector assignments are target specific and are typically done in the `INT_Initialize` function of `INT_XXX.S`. Nucleus uses the Decrementer (0x900) interrupt for its periodic timer interrupt.

Reentrancy

Reentrant C functions are functions that do not access or rely on global or static data. Nucleus PLUS services are fully reentrant. Diab Data library functions may or may not be reentrant, depending on the function. Please review the Diab Data documentation regarding the reentrancy of the Diab Data library functions



4

Nucleus PLUS Control Block Offsets

Task Control Block Offsets

HISR Control Block Offsets



Task Control Block (TCB) Offsets

The Task Control Block (TCB) contains information used to manage the execution of a task in the Nucleus PLUS system. This structure is defined in a manner that insures that most of its members are aligned on long-word boundaries.

Decimal Offset	HEX Offset	TCB Member
0	00	tc_created
12	0C	tc_id
16	10	tc_name
24	18	tc_status
28	1C	tc_delayed_suspend
32	20	tc_priority
36	24	tc_preemption
40	28	tc_scheduled
44	2C	tc_cur_time_slice
48	30	tc_stack_start
52	34	tc_stack_end
56	38	tc_stack_pointer
60	3C	tc_stack_size
64	40	tc_stack_minimum
68	44	tc_current_protect
72	48	tc_saved_stack_ptr
76	4C	tc_time_slice
80	50	tc_ready_previous
84	54	tc_ready_next
88	58	tc_priority_group
92	5C	tc_priority_head
96	60	tc_sub_priority_ptr
100	64	tc_sub_priority
104	68	tc_saved_status
108	6C	tc_signal_active
112	70	tc_entry
116	74	tc_argc
120	78	tc_argv
124	7C	tc_cleanup
128	80	tc_cleanup_info
132	84	tc_suspend_protect
136	88	tc_timer_active
140	8C	tc_timer_control
144	90	tc_signals
148	94	tc_enabled_signals
152	98	tc_signal_handler



HISR Control Block (HCB) Offsets

The HISR Control Block (HCB) contains information used to manage the execution of a HISR in the Nucleus PLUS system. This structure is defined in a manner that insures that most of its members are aligned on long-word boundaries.

Decimal Offset	HEX Offset	HCB Member
0	00	tc_created
12	0C	tc_id
16	10	tc_name
24	18	tc_not_used_1
28	1C	tc_not_used_2
32	20	tc_priority
36	24	tc_not_used_3
40	28	tc_scheduled
44	2C	reserved
48	30	tc_stack_start
52	34	tc_stack_end
56	38	tc_stack_pointer
60	3C	tc_stack_size
64	40	tc_stack_minimum
68	44	tc_current_protect
72	48	tc_active_next
76	4C	tc_activation_count
80	50	tc_entry





Nucleus C++

Introduction

**Building the Nucleus C++
Library From a DOS Prompt**

**Building and Executing the
C++ Demonstration System
From a DOS Prompt**

System Startup

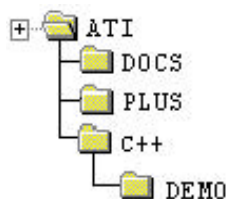
Operators New and Delete

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

Introduction

Nucleus C++ for the PowerPC 821/860 environment is shipped on CD-ROM. The CD contains all the C/C++ source code and include files for Nucleus C++, target specific assembly files, build and link command files, and an example system.

To install the Nucleus C++ software, execute `SETUP.EXE` from the distribution software. The Nucleus C++ library and demo rely on this structure and will not build correctly if this directory structure is modified. After installation, the directory structure should resemble the following:



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)

Building the Nucleus C++ Library From a DOS Prompt

Nucleus C++ is implemented as a library. The MS-DOS batch file `PLUSPLUS.BAT` contains the Diab Data assembler, compiler, linker, and archiver commands necessary to build the Nucleus C++ library. Execution of `PLUSPLUS.BAT` produces the Nucleus C++ library file `PLUSPLUS.LIB` in the `\O` subdirectory. This batch file will rebuild the PLUS library with an additional file.



Compiler Options

The compiler options used to build the Nucleus PLUS library are used. In addition, compilation parameters that are new to Nucleus C++ are as follows:

Compilation Flag	Meaning
-Xno-exception	Tells the compiler to disable exception handling.
-Xno-implicit templates	Tells the compiler to instantiate templates only where explicit instantiation syntax is used.
-Xno-rtti	Tells the compiler to disable run time type information.

Conditional Compilation

The Nucleus C++ library has several conditional compilation options. These conditional compilation flags enable various features within the Nucleus C++ library source. The flags are to be used in conjunction with the flags for the Nucleus PLUS kernel. They are defined using `#define` statements in the port specific header file `NUCPP.H`.

The conditional compilation options are as follows:

Compilation Flag	Meaning
<code>NU_ERROR_STRING</code>	Enables making an ASCII error string if a fatal system error occurs. This flag is applicable to the compilation of <code>ERD.C</code> , <code>ERI.C</code> , <code>ERC.C</code> , and the entire Nucleus C++ component.
<code>NU_NO_ERROR_CHECKING</code>	Disables error-checking shell on creation of the timer <code>HISR</code> in <code>TMI.C</code> . It is also applicable to the Nucleus C++ library component compilation. This will result in a significant run-time performance increase and will also reduce code size.
<code>NU_DEBUG</code>	Maps the data structures used by the application and defined in <code>NUCLEUS.H</code> to the actual internal data structures in Nucleus PLUS. This option must be used for Nucleus C++.



Building and Executing the C++ Demonstration System

Building an application is described in the "Getting Started" chapter of the *Nucleus C++ Reference* manual. Also, specific debugger commands are documented in the target specific notes for the Nucleus PLUS kernel supplied with your original documentation set.

Assuming that `NUCPP.H`, `PLUS.LIB` and `PLUSPLUS.LIB` are accessible from their respective directory, the following commands build and link the demonstration application.

```
DCC -v -g -c -tPPC860ES -D%_trg% -I..\ -I..\..\plus -o O\user_mmu.o
..\..\plus\user_mmu.c
DCC -v -g -c -tPPC860ES -Xno-exception -Xno-implicit-templates -I..\
-I..\..\plus -o O\demo.o demo.cpp

DLD -lc -m2 -tPPC860ES -o O\demo.elf O\CRT0.o O\int_%_trg%.o O\mmu.o
O\demo.o O\user_mmu.o ..\O\plusplus.lib ..\..\plus\O\plus.lib
..\..\plus\demo.lnk > O\demo.map
echo.
set _trg=
```

Typically, `*.S` and/or the `INIT.C` file of the Diab Data tools needs to be modified for a particular target environment.

The file `DEMO.CPP` contains the `InitializeCppApplication` function as well as all tasks and other objects. The `DEMO.LNK` file defines what needs to be linked together and where everything belongs for the corresponding board.

This batch file compiles `DEMO.CPP` and links it all together. The `DEMO.LNK` file contains information necessary for the linker. This file should be used as a template for any user-created linker command file.

A file named `DEMO.ELF` is produced by the batch file. Make sure all files are compiled with full debug information (`-g` option with the Diab compiler) Set up a configuration file that will initialize the board for SingleStep by doing the following:

- Go to the `\INIT` subdirectory of SingleStep and create or modify `SSTEP.INI`
- `SSTEP.INI` will contain one line that consists of the following:

```
{DIR}\CMD\{BOARD}.CFG
```

`{DIR}` is the directory where Single Step installed.

`{BOARD}` is the configuration filename. Look in the `\CMD` directory for the file that works.



Depending on the version of the SingleStep debugger, they should be similar to the following:

ESTSBC8X.CFG for the Motorola MPC860 EST board.
 860FADSFRZ.CFG for the Motorola MPC860 FADS board.
 MBX860.CFG for the Motorola MPC860 MBX board.

- Also required is a second file that needs to be present in the \CMD directory, along with the .CFG file. Again depending on the version of the SingleStep debugger, the file should be similar to one of the following.

ESTUPM.DBG for the Motorola MPC860 EST board.
 UPMFADS.DBG for the Motorola MPC860 FADS board.

NOTE: All of the files can be found on the distribution CD you received from ATI, in the folder \ATI\SDS, or on the internet at <http://www.sdsi.com>.

- Choose *File* and then select *Debug* in the SDS debugger to upload the O\DEMO.ELF executable to the target.
- Select the *Processor* tab, and select “*By Object File*”.
- Select the *Options* tab and uncheck “*Require Exact Symbol Names*”.
- The remaining options are checked. Click *OK* to upload.
- Click on the global icon to get a listing of all global variables in the system.
- Set a watch on the "screen" variable. It will appear as a text array in your *Watch* window. Each time you stop demo execution, the "screen" array will be updated and will appear similar to the screen output of the PLUS port if you had a serial driver. The array will display the status of the variables you normally watch with the PLUS port.

Please refer to Chapter 5 of the *Nucleus C++ Reference* manual to familiarize yourself with the demo application. This will explain good places to set breakpoints and serves as an excellent tutorial for the Nucleus C++ operating system.



System Startup

Nucleus C++ utilizes the Diab Data startup functions. Note: the application is not allowed to have a function named main.

INT_Initialize is responsible for setting up control registers, the interrupt vector table, several system stack areas, and typically the periodic timer interrupt. After this low-level initialization is complete, control is transferred to INC_Initialize for complete, high-level initialization of the Nucleus PLUS system.

INC_Initialize calls the Application_Initialize function for initialization of the Nucleus C++ components. The InitializeCppApplication function is called by Application_Initialize after all other initialization is complete, and before the Nucleus PLUS scheduler is invoked.

Note that before InitializeCppApplication is called, the Diab Data specific __init routine is called. This iterates the static constructor list. This routine is called after the Nucleus C++ system is initialized, supporting static RTOS objects. This support is new since the release of the *Nucleus C++ Reference* manual.



Operators New and Delete

The following Memory Management options are selected for the demonstration system running on the Software Development Systems Simulator. These options are selected in `NUCPP.H`.

Parameter	Meaning
<code>NU_HEAP_START_VALID</code>	Set to 1. <code>NU_HEAP_START</code> is a valid address.
<code>NU_HEAP_START</code>	Points to the beginning of the heap. Set to <code>first_available_memory</code> , which is initialized during startup.
<code>NU_HEAP_SIZE</code>	Specifies the number of bytes in the heap as 65536 or 64K Bytes of free store.
<code>NU_HEAP_REentrant</code>	Specifies whether malloc is reentrant. Not applicable for this port.
<code>NU_HEAP_SUSPEND</code>	Specifies how threads suspend during memory allocation if necessary. Possible values are <code>NU_NO_SUSPEND</code> , <code>NU_PRIORITY</code> , and <code>NU_FIFO</code> . Set to <code>NU_PRIORITY</code> by default.
<code>NU_HEAP_NUMBER_POOLS</code>	Specifies the number of "NU_HEAP_SIZE" byte memory pools created. Not applicable for this port.
<code>NU_HEAP_ZERO_MEMORY</code>	Specifies whether the new operator zeroes out the allocated memory before it is returned to the user. Set to TRUE by default.

Please refer to the Nucleus C++ Memory Management document for information on custom settings.



Nucleus FILE

Each chapter in this manual is specific to an individual Nucleus product.

Each product is purchased separately, and may not have been received with your particular distribution.

Introduction

Building the Nucleus FILE Library from a DOS Prompt

Building and Executing the FILE Demonstration System from a DOS Prompt

Architectural Considerations in PCDISK.H

Formatting A Disk

Device Driver Interface
DEVTABLE.C

Initializing the Nucleus PLUS Environment

Memory Allocation
PC_MEMORY.C

The RAMDISK Demonstration System

Introduction

Nucleus FILE is an MS-DOS compatible file system designed for embedded applications. Nucleus FILE is entirely written in ANSI C.

Nucleus FILE is implemented as a C library. It is combined with the Nucleus PLUS real-time multitasking kernel library to provide all file system services. Applications are compiled and then linked with the library. The resulting object may be downloaded to the target or placed in ROM.

Diab Data Development Tools provide a complete development environment for C and C++ applications targeted for the PowerPC family of microprocessors. The basic tools include a compiler, assembler, linker, and librarian.

Nucleus FILE for the PowerPC Diab Data development tools environment is shipped on CD-ROM. If any drivers have been purchased with Nucleus FILE, their source files are also shipped on the CD-ROM. To install the Nucleus FILE software, execute SETUP.EXE from the distribution software. After installation, the directory structure should resemble the following:



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)



Building the Nucleus FILE Library From a DOS Prompt

Nucleus FILE is implemented as a library. However, it requires the Nucleus PLUS library to be built before the Nucleus FILE object files are added. The MS-DOS batch file `FILE.BAT`, contains the compiler and librarian commands necessary to build the Nucleus FILE library. The Nucleus FILE library is built by executing the following command:

```
\ATI\FILE> file <cr>
```

The result will be the `FILE.LIB` file, and the Nucleus FILE object code will also be placed in the `FILE\O` directory.

Command Line Parameters

There are several command line parameters associated with compilation of each library element. The following is a description of each command line option for the Diab Data development tools on an MS-DOS host:

Command Line Option	Meaning
-c	Specifies an ".O" file, bypassing the link phase of the compiler.
-g	Generates symbolic debugger information with most optimizing turned off.
-r	Replaces the named ".O" files in the archive <code>FILE.LIB</code> .
-sR	Updates the symbol table in the archive file and sorts the COFF files in the archive.
-tPPC860ES	Specifies the target to compile to.
-v	Verbose output to the screen.

Conditional Compilation

The Nucleus FILE library has conditional compilation options. The conditional compilation flags enable the use of Nucleus PLUS with Nucleus FILE and allow the removal of Nucleus PLUS error checking. The conditional compilation options and their specification for the Diab Data development tools is as follows:

Compilation Flag	Meaning
-DPLUS	Indicates that Nucleus PLUS will be used in lieu of Nucleus RTX.
-DNU_NO_ERROR_CHECKING	This option excludes error checking in Nucleus PLUS.



Building and Executing the FILE Demonstration System From a DOS Prompt

Building an application is dependent on the drivers that you are using. Nucleus FILE is supplied with a specific demonstration program for all systems, this demonstration uses RAM Disk for the driver level capabilities. The program can be executed on any PowerPC target platform.

The subdirectory `\FILE\RAMDEMO` created with your release contains a demonstration program for executing a RAM Disk version of the file system on a PowerPC target platform. This section describes the installation and execution of the demonstration program. Use this demonstration program to familiarize yourself with the capabilities and usage of Nucleus FILE. It covers a broad range of interfaces including formatting a disk, creating, deleting, and traversing directories, and writing, reading, renaming, and deleting files. The demonstration system is built using the file, `RAMDEMO.BAT`.

NOTE: Be sure to check the file, `RAMDEMO.BAT` and make sure that the `_trg` value is set to the board that applies to you.

Considering the FILE library is built, you must build the demonstration program. The demonstration program is built by executing the following command at the DOS prompt:

```
\ATI\FILE\RAMDEMO> ramdemo <cr>
```

The `RAMDEMO.BAT` file compiles all of the necessary modules for the RAM Disk demonstration and includes them in the `FILE.LIB` library file. Note that many of the generic Nucleus FILE object files are replaced with RAM Disk demonstration versions. Pay close attention to these files when you begin to develop your own application.

After building the demonstration system, you can execute it on the target platform. The following sections describe procedures for loading the PowerPC version of Nucleus PLUS and Nucleus FILE (with the RAMDISK Demonstration) to the SingleStep Simulator. After loading the program, if your target permits, you will see some statistics printed on the display periodically. Please examine the `RAMDEMO.C` source code to understand what is being displayed.

Please note that a warning may result due to the re-declaration of `__init_main` in `INT.O` for Nucleus PLUS and `INIT.O` for the Diab Data tools. The label `__init_main` may be removed from one of these modules to avoid this warning. However, this warning does not interfere with `RAMDEMO`'s operation.

Using the SingleStep Debugger

An executable file named `RAMDEMO.ELF` is produced by the `RAMDEMO.BAT` batch file. This file is in ELF format and is ready for execution on the Single Step Debugger. The following steps may be taken to load and run the Nucleus FILE demonstration application:



Load the debugger from within Windows.

Set up the Debugger for execution of the program by selecting to debug the `RAMDEMO.ELF` program that was created by `RAMDEMO.BAT`.

The debugger is now ready, and the `RAMDEMO.ELF` program can be executed by selecting *Go*.

Examine the following counters to verify that the `RAMDEMO` program is running correctly.

```
RAM_Files_Created
RAM_Directories_Created
RAM_Directories_Deleted
RAM_Directories_Scanned
RAM_Files_Scanned
RAM_Files_Renamed
RAM_Files_Deleted
```

The RAM Disk demonstration program was originally designed to be interactive. However, because many embedded systems do not have standard I/O (or have I/O in a non standard form), we have provided two `#define` macros for activating (or deactivating) input and output I/O. To activate the macros, define them as 1, otherwise define them as 0. The Diab Data tools version of the `RAMDEMO` program is shipped with the output (`PRINT_OUT`) deactivated, and the input (`INPUT_OK`) deactivated. Please note that the input routines are port specific and should be modified unless you have purchased one of Accelerated Technology's serial driver templates.

Architectural Considerations PCDISK.H file

Nucleus FILE was originally developed for the Intel architecture. Although it was designed to be, and is, portable, there are some differences between the Intel architecture and most other architectures. To accommodate this difference, the `#DEFINE INTEL` is included in the file `PCDISK.H`. You must ensure that it is set to 0 for the PowerPC implementation of Nucleus FILE. The version of `PCDISK.H` found in the `\FILE` directory includes this change. However, the standard version of `PCDISK.H` found in the `\FILE\TEMPLATE` directory does not. Please ensure that this change is made before you begin building a version of the file system for your application.

Formatting A Disk

Nucleus FILE is compatible with MS-DOS 4.0 and below. Therefore, a facility is provided to format a disk in MS-DOS format. The formatting is performed by the routine `NU_Format`. See the Nucleus FILE reference manual for details on the use of this routine. Also, study the RAM Disk demonstration program carefully to determine which parameters must be set up for your formatting operation.



Device Driver Interface DEVTABLE.C

The file `DEVTABLE.C` contains declarations that are used by Nucleus FILE to associate device drivers with the file system. In the directory `\FILE\TEMPLATE` there is a complete version of `DEVTABLE.C`. It is based on a system configuration with a hard disk, floppy disk, and RAM disk. There are multiple configurations for each. Many of these interfaces are based on Accelerated Technology's own drivers. You should examine this file to become familiar with the use of the device driver interface. A simplified version of the file is provided in the `\FILE` directory of your release. This is also a good file to examine, as it may be closer to your implementation.

Initializing the Nucleus PLUS Environment NUFI.C

Nucleus FILE requires various Nucleus PLUS resources. These resources include memory allocation, task initialization, and semaphore allocation as well as task pointers. Because Nucleus FILE can accommodate the Nucleus PLUS application, certain conversions are also necessary. These Nucleus PLUS initialization functions and conversions are provided in the files `NUFI.C` and `NUFP.C`. These files warrant close examination so that you can develop a fuller understanding of the file system.

Of note are the special versions of `NUFI.C` and `NUFP.C` in the `\FILE` directory. These files have been modified to accommodate the `RAMDISK` demonstration program. This has been done by activating additional tasks and reducing the number of FAT allocations. Again, close review of these files are profitable for better determining the requirements for your application.

Memory Allocation PC_MEMORY.C

Nucleus FILE requires memory for a number of data structures and buffers. Although most of these allocations remain constant, some of them are dependent on the number of disks in the system. Each disk requires a FAT buffer, which is allocated in `PC_MEMORY.C`.

In addition to the copy of `PC_MEMORY.C` found in the `\FILE\TEMPLATE` directory, a version of `PC_MEMORY.C` in the `\FILE` directory is set up to use only one disk and consequently only one FAT. Both of these files are worthy of examination to familiarize you with the memory allocation requirements of Nucleus FILE.



The RAMDISK Demonstration Program

Interpreting RAMDEMO.C

The RAMDEMO.C file provides a comprehensive look at using the file system. It contains code to format a disk, write and delete file, and make, traverse, and delete directories. Please study it carefully before embarking on your own development efforts.

RAMDEMO Initialization

As with all Nucleus PLUS programs, the tasking environment for RAMDEMO.C must be initialized. The tasks that are used in the demonstration program are:

```
void    nuf_task_0(UNSIGNED argc, VOID *argv);
void    nuf_task_1(UNSIGNED argc, VOID *argv);
void    nuf_task_2(UNSIGNED argc, VOID *argv);
void    display_task(UNSIGNED argc, VOID *argv);
```

These tasks are created in the routine Application_Initialize in the file NUFI.C. Please review this file for a better understanding of the system initialization requirements for Nucleus FILE.

Registering Tasks as File System Users

By examining any of the tasks in RAMDEMO.C you can observe the process that must be undertaken to register a task as a file system user. This is necessary to provide for multi-tasking access to the file system. The following code fragment extracted from one of the tasks in RAMDEMO.C illustrates the registration of the task with the file system. Note that bail_out is a function local to RAMDEMO.C.

```
/* Each task must register as a Nucleus user */
    if (!NU_Become_File_User())
        bail_out(CANT_REGISTER);
```

In addition to registering as a user of the file system, a disk must be opened for use by the task. The following code fragment extracted from RAMDEMO.C illustrates this process.

```
/* Prepare the disk for use by this task. */
    if (!NU_Open_Disk(__RAM))
        bail_out(CANT_OPEN_DISK);
```



Input and Output for RAMDEMO.C

In the first portion of RAMDEMO.C two manifest constants are provided, INPUT_OK and OUTPUT_OK. RAMDEMO.C was developed to be generic across all Nucleus platforms. Some platforms do not afford the capability of input and output. Both the INPUT_OK and OUTPUT_OK manifest constants are defined as 0 and therefore no input or output is available in the default configuration.

Restriction

RAMDISK.O is the driver for RAMDEMO.C. You can see the invocation of its routines in the file DEVTABLE.C. We provide for a maximum RAM Disk size of 64Kbytes with the standard release of Nucleus FILE. An unrestricted version of the RAM Disk driver is available from Accelerated Technology. We have configured the RAM Disk size by initializing the variable NUM_RAMDISK_PAGES to 16. This provides for 64Kbytes because each page holds 4Kbytes. If you try to increase NUM_RAMDISK_PAGES beyond 16, the system will generate an error and will not function properly.



Nucleus NET

Introduction

Building the Nucleus NET Library From a DOS Prompt

Overview of The Demonstration System

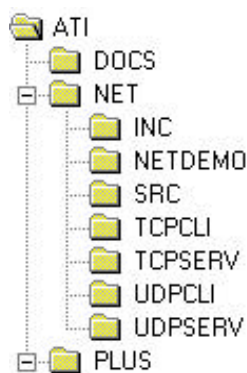
Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.



Introduction

This chapter discusses the installation of the Nucleus NET source code, building the NET library and building the demonstration applications. Nucleus PLUS must be installed and working properly, before Nucleus NET is installed. It is recommended that you build and execute the Nucleus PLUS demonstration application prior to building Nucleus NET. The building of the Nucleus PLUS library is discussed in the chapters '*Nucleus PLUS*' and '*Nucleus PLUS Usage*' of this manual. Instructions for building the library do not apply to Nucleus EDE ports. Please refer to your Nucleus EDE Online Help and Chapter 22 for more information.

To install the Nucleus NET software, execute `SETUP.EXE` from the distribution software. All Nucleus products associated with this Motorola PowerPC 821/860 port using the Diab Data Tools must be installed in a common directory (`C:\ATI` by default). NET will require PLUS to be present in this directory structure. NET expects to be within this directory structure. After installation, the directory structure should resemble:



If you purchased a driver (Ethernet, SLIP, PPP, etc.) from Accelerated Technology, you should refer to that chapter of the manual for specific instructions. If no driver was purchased you can build the Nucleus NET library and Nucleus NET demo applications, but you will be unable to execute the demo applications.

There are several files required from the Nucleus PLUS installation. Verify that `NUCLEUS.H` is located in the corresponding `\PLUS` directory, and `PLUS.LIB` is located in the `\PLUS\O` directory.



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)

Building The Nucleus NET Library From a DOS Prompt

Once the Nucleus NET files have been installed, and the driver (if any) has been installed, you will be ready to build the Nucleus NET library, `NET.LIB`.

Three boards are supported, listed alphabetically, the Embedded Systems Tools (EST) SBX8XX, the Motorola FADS 821/860, the Motorola MBX 821/860.

The MS-DOS batch file contains the Diab Data assembler, compiler, and librarian commands necessary to build the Nucleus NET library. To build the NET library, simply enter the following command at an DOS prompt.

```
\ATI\NET> net <cr>
```

Execution of the file `NET.BAT` will produce the `O\NET.LIB` library file. It is assumed that interrupts will be used.

Compilation Switches Used

There are several compilation switches that are used to create each object file. The following lists the compilation switches used for the Diab Data compiler:

Compiler Switch	Meaning
<code>-c</code>	Specifies a compile to <code>.OBJ</code> without linking.
<code>-v</code>	Specifies that source level debugging is on.
<code>-g</code>	Specifies to add debugging information.
<code>-tPPC860ES</code>	Create code for PowerPC 860 ELF format.



Defines Used for Compilations

Nucleus NET library has several conditional compilation options. Conditional compilation requests are specified with the `-D` compilation option. The following conditional compilation flags enable various features within the Nucleus NET library.

Compilation Flag	Description
INTERRUPT	Enables compilation of Nucleus NET for interrupt mode. Use of Nucleus NET in polling mode is not recommended, as performance will be very poor. Also, not all drivers provided by Accelerated Technology support polling mode.
PLUS	Enables compilation of Nucleus PLUS calls only. If PLUS is not defined the Nucleus RTX service calls will be used. (NOTE: Nucleus RTX is no longer supported with Nucleus NET.)
PACKET	If PACKET is defined, packets are transmitted as soon as they are ready, otherwise they are placed in a queue when they are ready. Use of Nucleus NET in polling mode is not recommended, as performance will be very poor. Also, not all drivers provided by Accelerated Technology support polling mode.

Note 1) The `INTERRUPT` flag will affect only some of the network files as these files are the only ones whose behavior varies with the mode of operation. The files that are affected are:

ARP.C	EQUEUE.C	NET.C
TCP.C	TCPVARS.C	TOOLS.C
USER.C	Any Driver	

Note 2) The `PACKET` flag will only affect `NET.C` and the driver files.

Note 3) The `PQUICC` driver only supports the `NON-PACKET` mode of transmission.



TARGET.H

TARGET.H is target specific header file that contains configuration information for Nucleus NET. The file is well commented. Please read the comments within TARGET.H for more information about the options it contains.

Nucleus NET Interface

For a discussion of the Nucleus NET interface routines, see the *Nucleus NET Reference Manual*.

Overview of Demonstration System

Nucleus NET contains four demos. The demo shows either the TCP or UDP protocol working as a server or a client. The demos are designed to communicate with the Winsock application \NET\NETDEMO\NETDEMO.EXE which runs on any Windows 95/98 or Windows NT machine connected to the network. NETDEMO.EXE can be run in 4 different modes so that it can communicate with any of the embedded demos. The two applications will communicate over the echo port which is port 7. The server echos back data sent to it by the client. The demos shipped with NET are compile only. They require a driver that can be written by the user or purchased from Accelerated Technology Inc. If a driver was purchased from ATI, build and run those demos instead of these NET demos. However, the functionality of the demos are the same. Please read this section and the chapter regarding the driver purchased.

Using the Nucleus NetDemo Application

The main screen for the Nucleus NetDemo Application is shown in Figure 7.1. There are three fields that are of interest. The fields are *Embedded Transfer Information*, *Winsock Status*, and *Test Applications*.



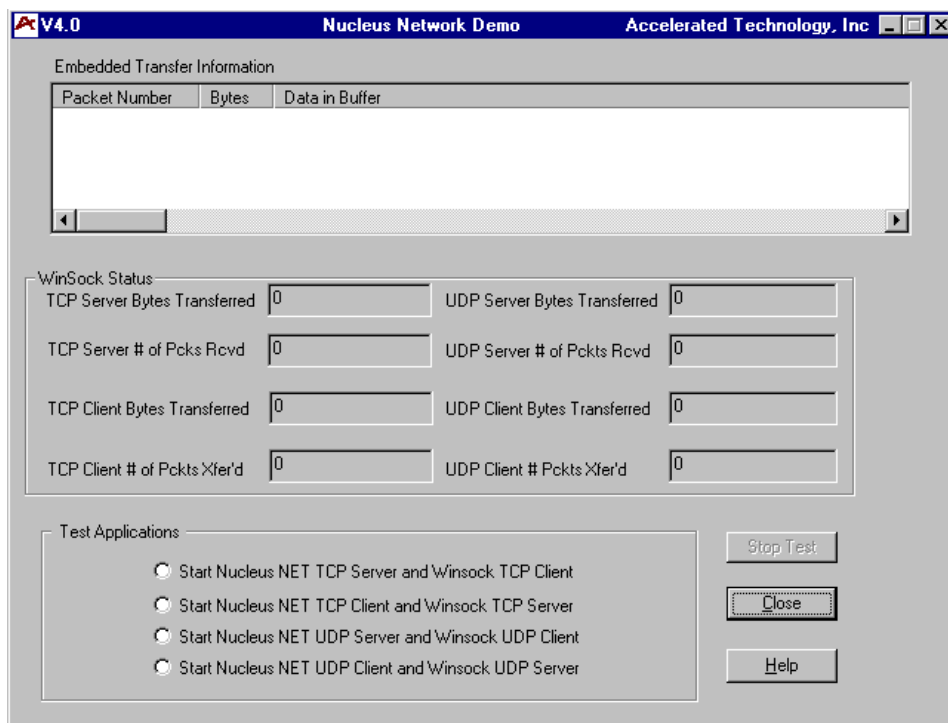


Figure 7.1 The NetDemo Main Screen

The *Embedded Transfer Information* field holds information that was received by the Winsock `Recv` command issued from the Nucleus NET application. The information is divided into three parts, *Packet Number*, *Number of Bytes in Packet* and the *Data Buffer*. The Scroll Bar may be used to view all of the information. The test application will accept 2000 packets, and will continue to accept more packets. It will, however, start to remove the first packet from sight, and continue to add the latest packet at the bottom of the scroll status.

The *Winsock Status* shows the status of the Winsock application that is running during each test. The information will display “total bytes sent/received”, and “total # number packets sent/received”, depending on which TEST is being performed.

The bottom section of Figure 7.1 shows the *Test Applications*, which consists of *Start TCP Server*, *Start TCP Client*, *Start UDP Server*, and *Start UDP Client*. To perform each of these test applications, refer to the embedded Nucleus NET applications that were provided with your Nucleus NET code.

Settings Common to all the NET Demos

All of the NET demos include code to set up the driver. The code is different for each driver but each driver will have the same code in all four demos. Here is a listing of driver initialization code from a compile only NET demo:

```
devices[0].dv_name = "DRIVER_0";
devices[0].dv_hw.ether.dv_irq = DEMO_IRQ;
devices[0].dv_hw.ether.dv_io_addr= DEMO_IO_ADDR;
devices[0].dv_hw.ether.dv_shared_addr = 0;
devices[0].dv_init = Init;
devices[0].dv_flags = 0;
memcpy (devices[0].dv_ip_addr, cli_ip_addr, 4);
memcpy (devices[0].dv_subnet_mask, subnet, 4);
```

Each demo also has a `printf` symbol that is defined. This function is for debugging purposes. The default is a function that is stubbed out and gives no debugging info. `printf` can be set to the name of a user supplied debugging function.

TCP Server Demo

`TCPSESV.C` is the TCP Server Demo. It accepts a limited number of connections from clients and echos the data the client sends. Before building the demo modify the following lines of code:

```
#define DEMO_IRQ          5
#define DEMO_IO_ADDR      0x0300L
#define DEMO_MAX_CONNECTIONS 10
#define DEMO_SERVER_IP_ADDR {100,200,300,400}
```

`DEMO_IRQ` and `DEMO_IO_ADDR` are driver specific defines that are not used in this compile only demo. `DEMO_MAX_CONNECTIONS` is the number of pending connections the server will accept. `DEMO_SERVER_IP_ADDR` is the IP address of the board.

Use this command line to build the TCP Server demo. Remember, each driver contains its own demos so the path the batch file that builds each demo may be slightly different.

```
\ATI\NET\TCPSESV> tcpsevr <cr>
```

This demo is compile only unless a driver is supplied. If a driver is supplied then `\ATI\NET\TCPSESV\O\TCPSESV.ELF` is downloaded to the board just like the PLUS demo. For more details about downloading code see Chapter 2. Always start the server application first. This means start the embedded application before `NETDEMO.EXE`.



Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

Once NETDEMO.EXE is started, click on the 'Start TCP Server Application'. The Pop-up window shown in Figure 7.2 below will appear. This will allow you to type in the IP address of the Nucleus NET TCP Server. After inputting your specific IP address, click *OK*.

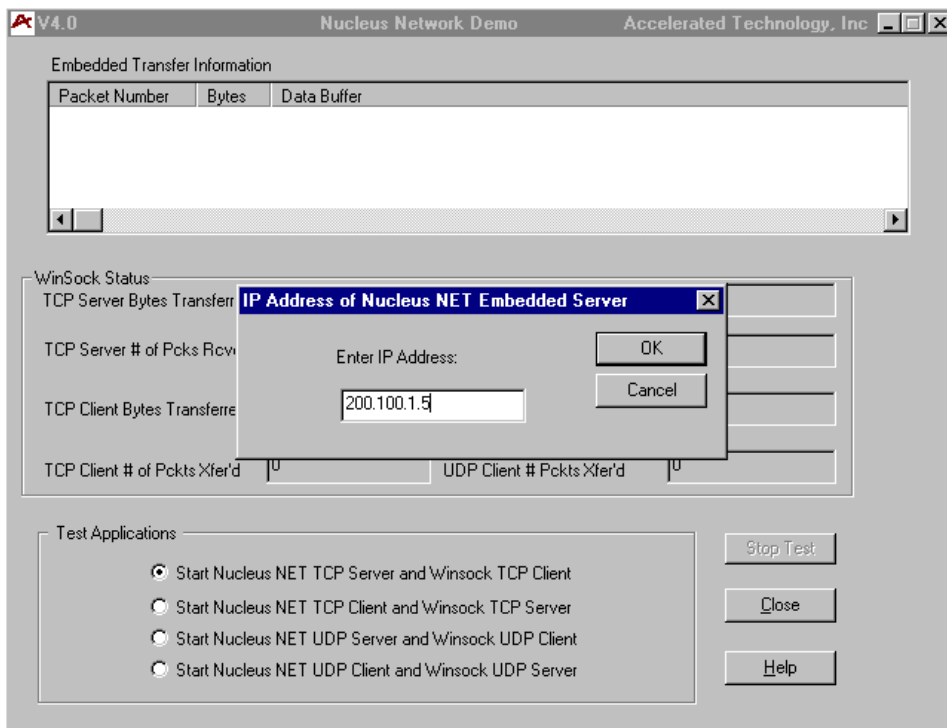


Figure 7.2 Configuring the Nucleus NET TCP Server



The next window will be displayed as shown in Figure 7.3. This Pop-up window will display a message to start the Nucleus NET embedded TCP Server. Once the embedded TCP Server is started click *OK*.

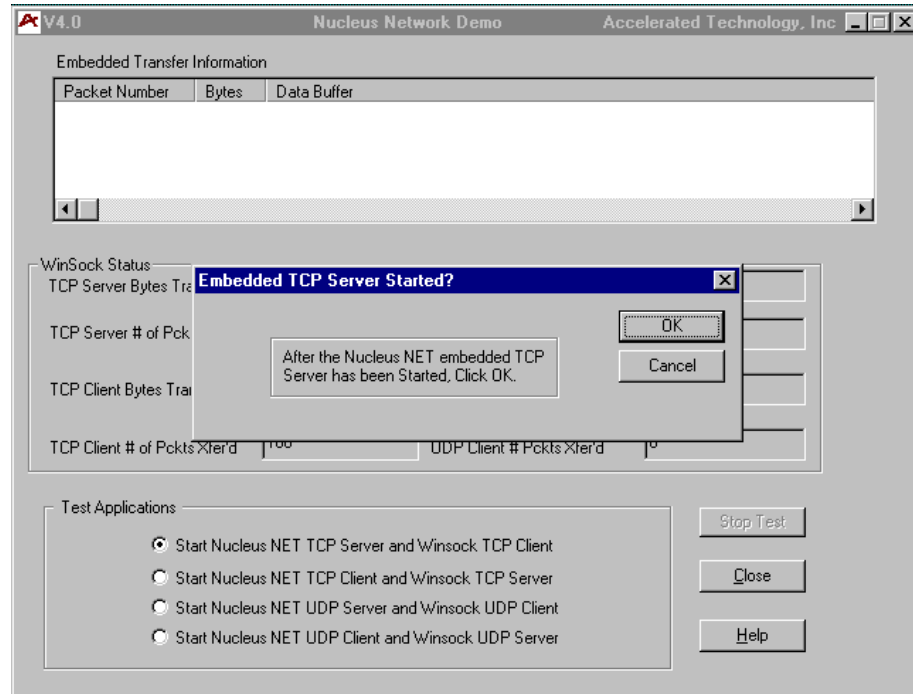


Figure 7.3 Starting the Nucleus NET TCP Server

Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

You should configure the Winsock TCP Client as shown in Figure 7.4. This Pop-up window allows you to enter the *Size in Bytes* of the TCP client packet, and the *Number of transfers* that the TCP Client packet will be transferring. Click *OK* , after you have entered the values.

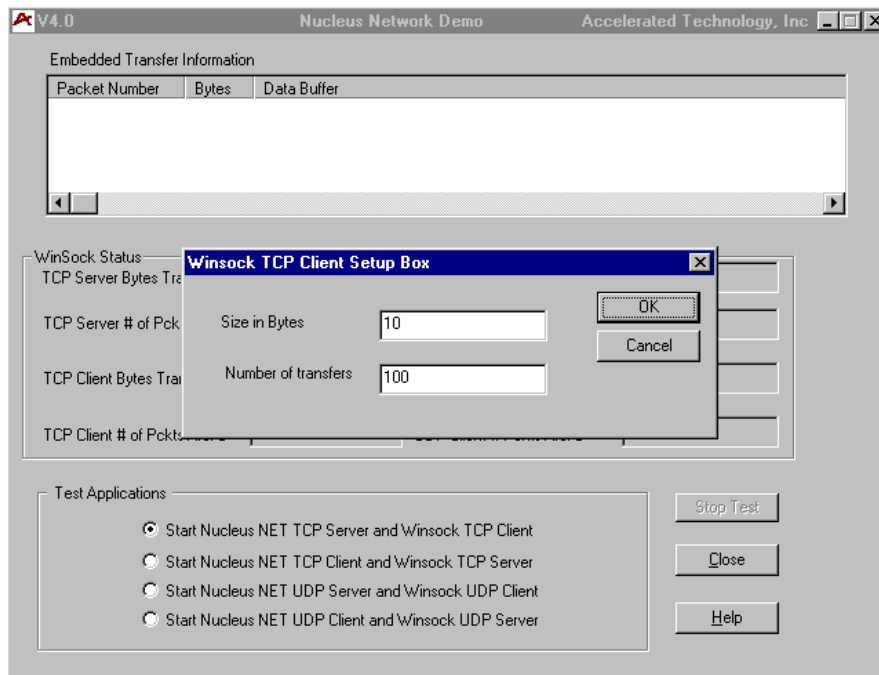


Figure 7.4 The Size in Bytes/Number of Transfers Screen



Data should be transferring at this time and you should see status being updated. The screen should appear similar to the screen displayed in Figure 7.5.

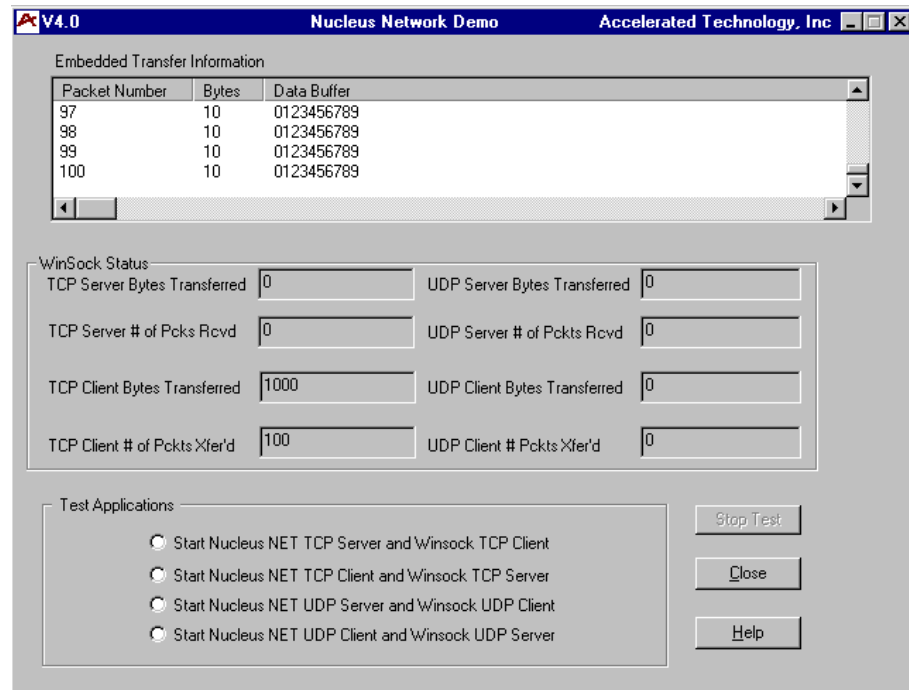


Figure 7.5 TCP Server/Client Status Update Screen

TCP Client Demo

TCPCLI.C is the TCP Client Demo. It connects to a server and sends *"This is TCP test packet # X"* where X is the number of packets that have been sent. Before building the demo modify the following lines of code:

```
#define printf          PRINTF
#undef  USE_HOST_TABLE
#define DEMO_IRQ        5
#define DEMO_IO_ADDR    0x0300L;
#define DEMO_SERVER_IP_ADDR {100,200,300,400}
#define DEMO_CLIENT_IP_ADDR {100,200,300,400}
#define DEMO_SERVER_NAME "server_name"
```



Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

If `USE_HOST_TABLE` is defined the demo will look for the servers IP using a call to `NU_Get_Host_By_Name`, otherwise the IP address must be supplied by the user. If `USE_HOST_TABLE` is defined there must be a machine to help resolve the DNS or the name must be in `HOST.C` before the `NET` library is compiled. `DEMO_IRQ` and `DEMO_IO_ADDR` driver specific defines that are not used in this compile only demo. `DEMO_SERVER_IP_ADDR` needs to be set to the IP address of the computer running `NETDEMO.EXE` and the name of that computer should be assigned to `DEMO_SERVER_NAME`. The IP address of the board should be assigned to `DEMO_CLIENT_IP_ADDR`.

Use this command line to build the TCP Client Demo. Remember, each driver contains its own demos so the path and the batch file that builds each demo may be slightly different.

```
\ATI\NET\TCPCLI> tcpcli <cr>
```

This demo is compile only unless a driver is supplied. If a driver is supplied then `NET\TCPCLI\O\TCPCLI.ELF` is downloaded to the board just like the `PLUS` demo. For more details about downloading code see Chapter 2. Always start the server application first. This means execute `NETDEMO.EXE` before starting the embedded demo.



Once NETDEMO.EXE is started, click on the 'Start TCP Client' button. This will display a Pop-up window as shown in Figure 7.6. Click *OK*. Then start the Nucleus NET embedded TCP client. Once *OK* is clicked, the Winsock TCP server is started. This insures that TCP client data is not being transmitted before the Winsock TCP server is prepared to accept data.

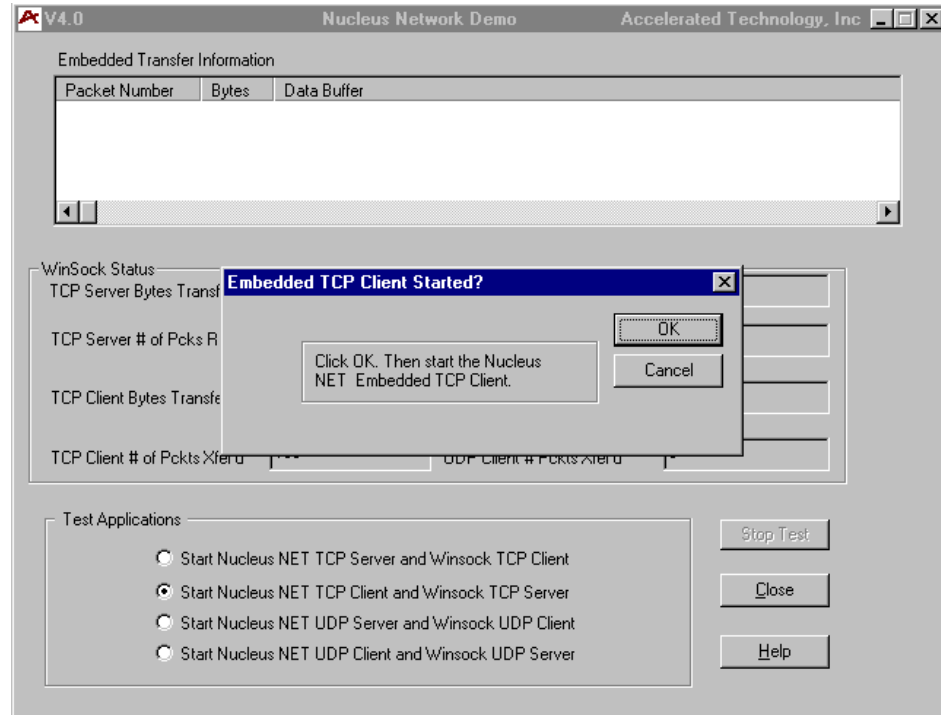


Figure 7.6 Starting the TCP Client

Figure 7.7 is an example of how the screen should appear while the test is running.

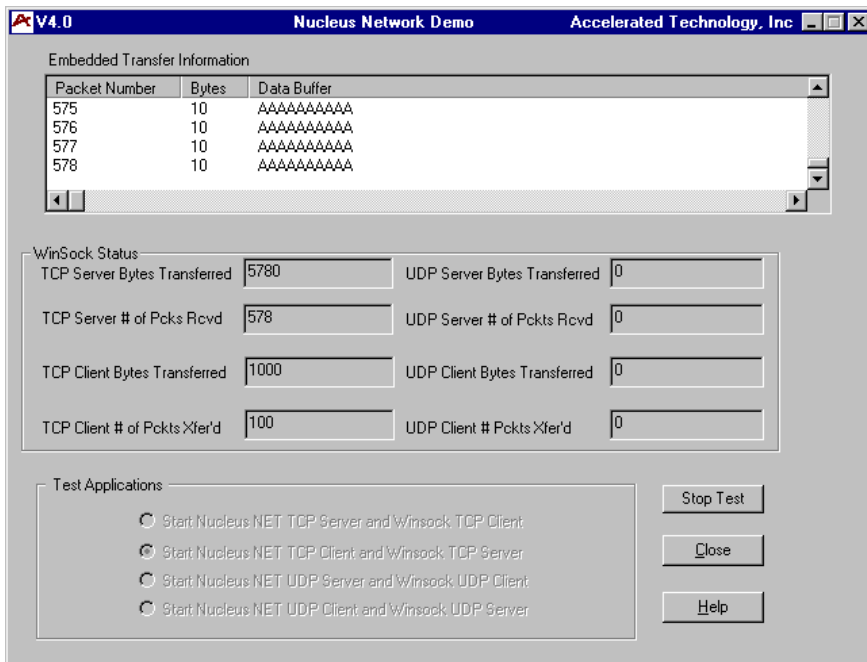


Figure 7.7 Testing the TCP Client Transfer Rate

UDP Client Demo

UDPCLI.C is the UDP Client Demo. It connects to a server and sends packets containing strings of capital As. Before building the demo modify the following lines of code:

```
#define DEMO_IRQ          5
#define DEMO_IO_ADDR      0x0300L
#define DEMO_CLIENT_IP_ADDR {100,200,300,400}
#define DEMO_SERVER_IP_ADDR {100,200,300,400}
#define DEMO_SERVER_NAME  "server_name"
```



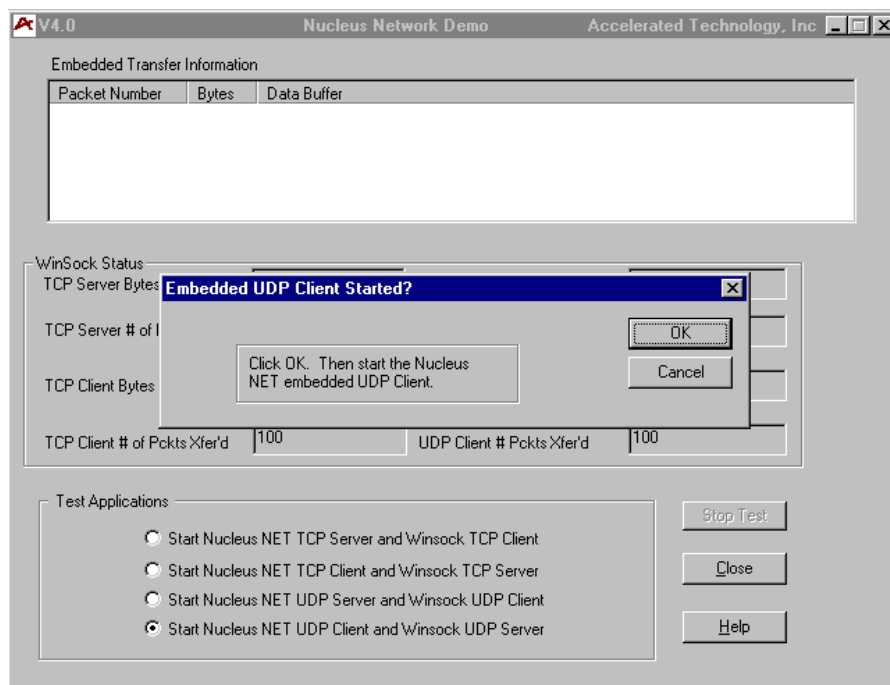


Figure 7.8 Starting the UDP Client

DEMO_IRQ and DEMO_IO_ADDR are driver specific defines that are not used in this compile only demo. The settings are very similar to those used in TCPCLI.C. DEMO_SERVER_IP_ADDR needs to be set to the IP address of the computer running NETDEMO.EXE and the name of that computer should be assigned to DEMO_SERVER_NAME. The IP address of the board should be assigned to DEMO_CLIENT_IP_ADDR.

Use this command line to build the UDP Client Demo. Remember, each driver contains its own demos so the path the the batch file that builds each demo may be slightly different.

```
\ATI\NET\UDPCLI> udpcli <cr>
```

This demo is compile only unless a driver is supplied. If a driver is supplied then \ATI\NET\UDPCLI\O\UDPCLI.ELF is downloaded to the board just like the PLUS demo. For more details about downloading code see Chapter 2. Always start the server application first. This means execute NETDEMO.EXE before starting the embedded demo.

Once NETDEMO.EXE is started, click on the 'Start UDP Client' button. This will display a Pop-up window as shown in Figure 7.8. Click *OK*. Then start the Nucleus NET embedded UDP client. Once OK is clicked, the Winsock UDP server is started. This insures that UDP client data is not being transmitted before the Winsock UDP server is prepared to accept data.



An example of how the screen should appear while the test is running, is shown below in Figure 7.9.

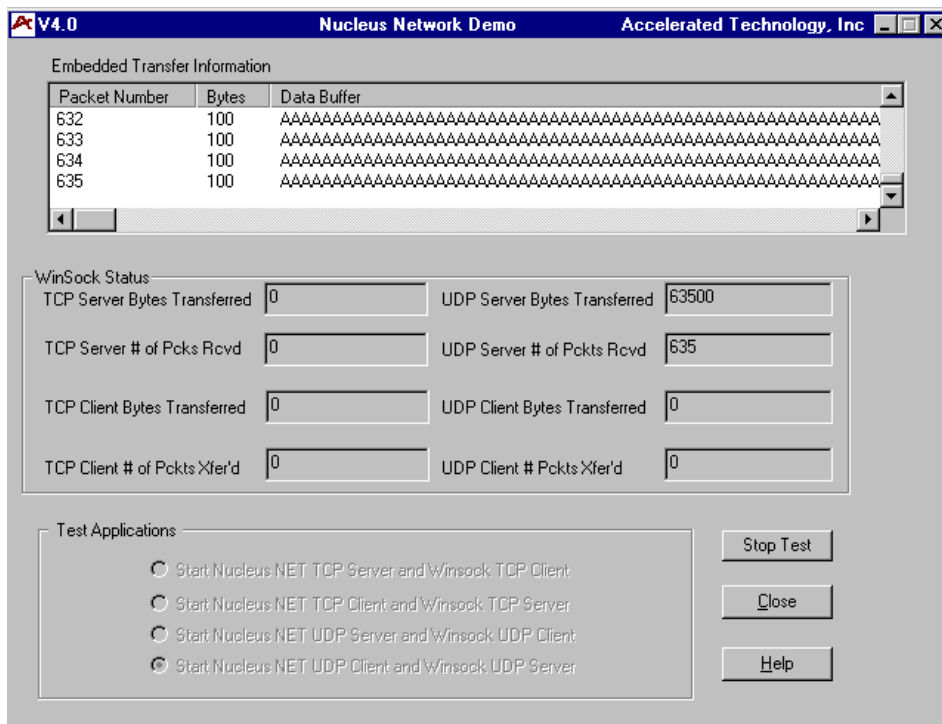


Figure 7.9 Testing UDP Client Transfer Rate

UDP Server Demo

UDPSERV.C is the UDP Server Demo. It accepts connections from clients and echos the data the client sends. Before building the demo, modify the following lines:

```
#define DEMO_IRQ          5
#define DEMO_IO_ADDR      0x0300L
#define DEMO_CLIENT_IP_ADDR {100,200,300,400}
#define DEMO_SERVER_IP_ADDR {100,200,300,400}
#define DEMO_CLIENT_NAME  "client_name"
#define DEMO_SERVER_NAME  "server_name"
```

DEMO_IRQ and DEMO_IO_ADDR are driver specific defines that are not used in this compile only demo. DEMO_CLIENT_ADDR needs to be set to the IP address of the PC running NETDEMO.EXE. DEMO_CLIENT_NAME is the name of this computer running NETDEMO.EXE. DEMO_SERVER_IP_ADDR needs to be set to the IP address of the embedded board. DEMO_SERVER_NAME is the name of the board.



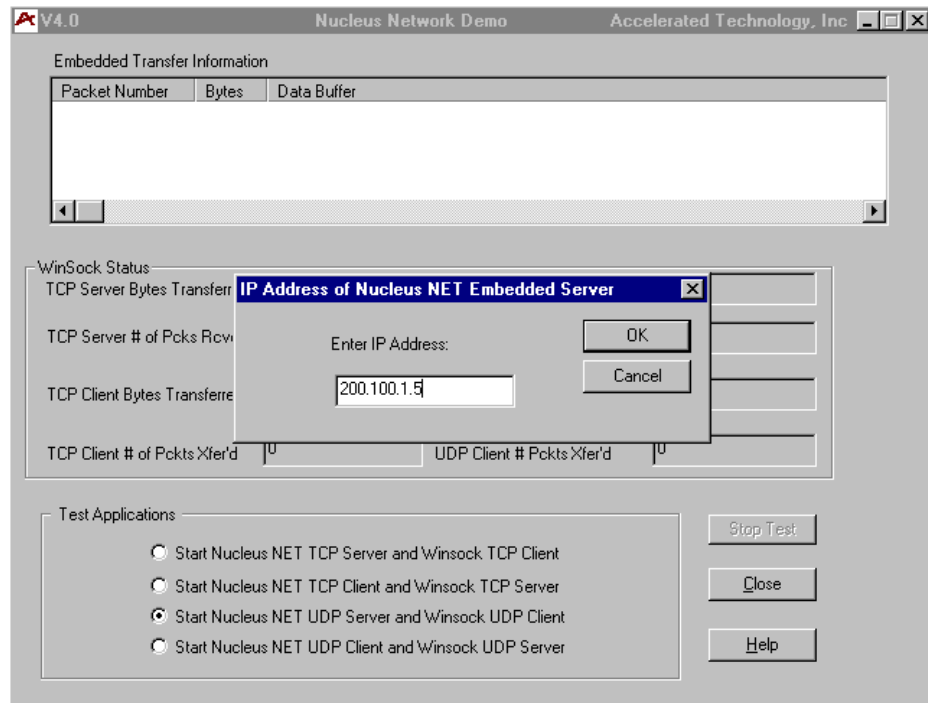


Figure 7.10 Starting the UDP Server Application

Use this command line to build the UDP Server Demo. Remember, each driver contains its own demos so the path and the batch file that builds each demo may be slightly different.

```
\ATI\NET\UDPSERV> udpserv <cr>
```

This demo is compile only unless a driver is supplied. If a driver is supplied then \NET\UDPSERV\O\UDPSERV.ELF is downloaded to the board just like the PLUS demo. For more details about downloading code see Chapter 2. Always start the server application first. This means start the embedded application before NETDEMO.EXE.

Once NETDEMO.EXE is started, click on 'Start UDP Server Application'. The pop-up window shown in Figure 7.10 below will appear. This will allow you to type in the IP address of the Nucleus NET UDP Server. After inputting your specific IP address, click OK.

Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

The next window, shown in Figure 7.11, will be displayed. This Pop-up window will display a message to start the Nucleus NET embedded UDP Server. Once the embedded UDP Server is started, click *OK*.

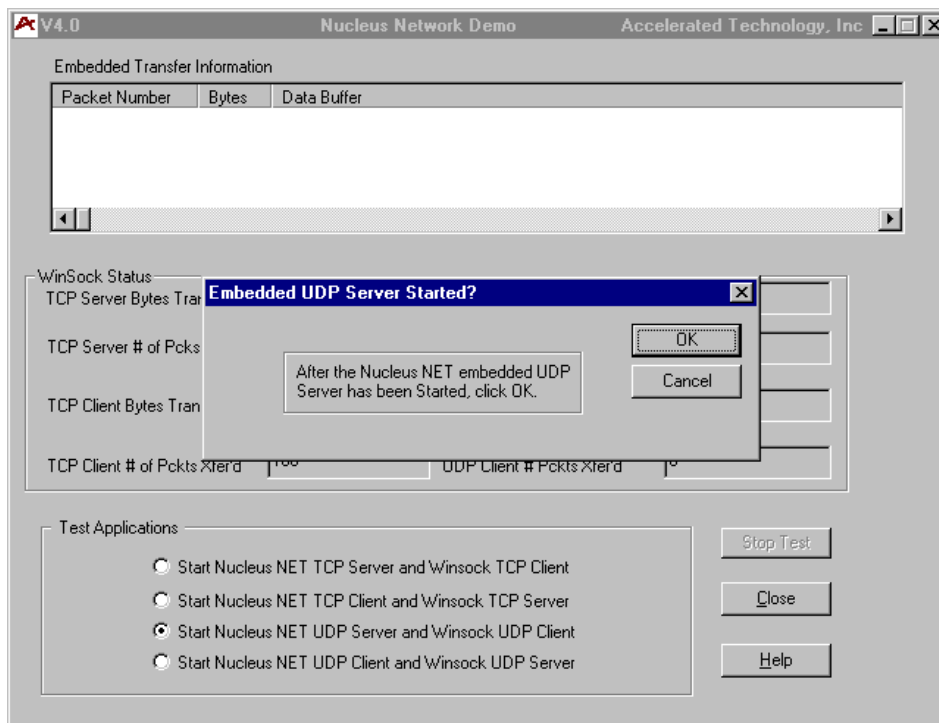


Figure 7.11 The UDP Server Start message



The next Pop-up window is displayed, which will allow you to configure the Winsock UDP Client and is shown in Figure 7.12 below. This Pop-up window will allow you to enter the *Size in Bytes* of the UDP client packet, and the *Number of Times to Transfer* by the UDP Client packet. Click *OK* after you have entered the appropriate values.

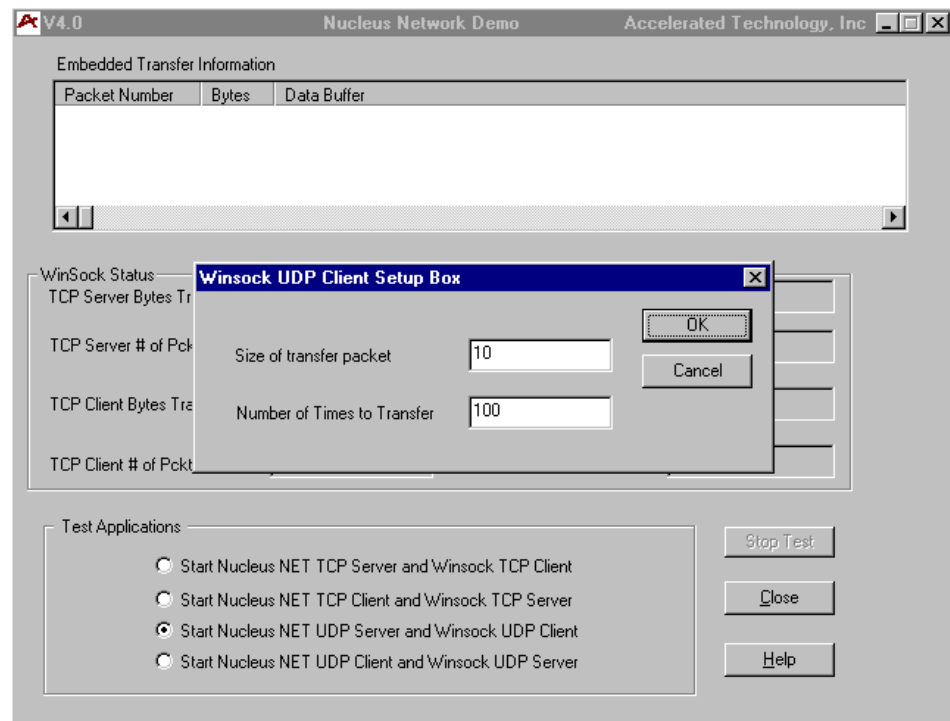


Figure 7.12 Size of Packets/Number of Times to Transfer Screen

Data should be transferring at this time, and you should see status being updated. The screen should look similar to Figure 7.13.

The screenshot shows a window titled "Nucleus Network Demo" by Accelerated Technology, Inc. The window is divided into three main sections. The top section, "Embedded Transfer Information", contains a table with four rows of data. The middle section, "WinSock Status", displays various statistics for both TCP and UDP connections. The bottom section, "Test Applications", provides radio button options for starting different network tests. On the right side of the bottom section are three buttons: "Stop Test", "Close", and "Help".

Packet Number	Bytes	Data Buffer
97	10	0123456789
98	10	0123456789
99	10	0123456789
100	10	0123456789

WinSock Status

TCP Server Bytes Transferred	5780	UDP Server Bytes Transferred	0
TCP Server # of Pckts Rcvd	578	UDP Server # of Pckts Rcvd	0
TCP Client Bytes Transferred	1000	UDP Client Bytes Transferred	1000
TCP Client # of Pckts Xfer'd	100	UDP Client # Pckts Xfer'd	100

Test Applications

- ☐ Start Nucleus NET TCP Server and Winsock TCP Client
- ☐ Start Nucleus NET TCP Client and Winsock TCP Server
- ☐ Start Nucleus NET UDP Server and Winsock UDP Client
- ☐ Start Nucleus NET UDP Client and Winsock UDP Server

Stop Test
Close
Help

Figure 7.13 UDP Server/Client Transfer Status Information

Stopping the Test Application

To stop the test application while running, click the *Stop Test* button as shown in Figure 7.14.

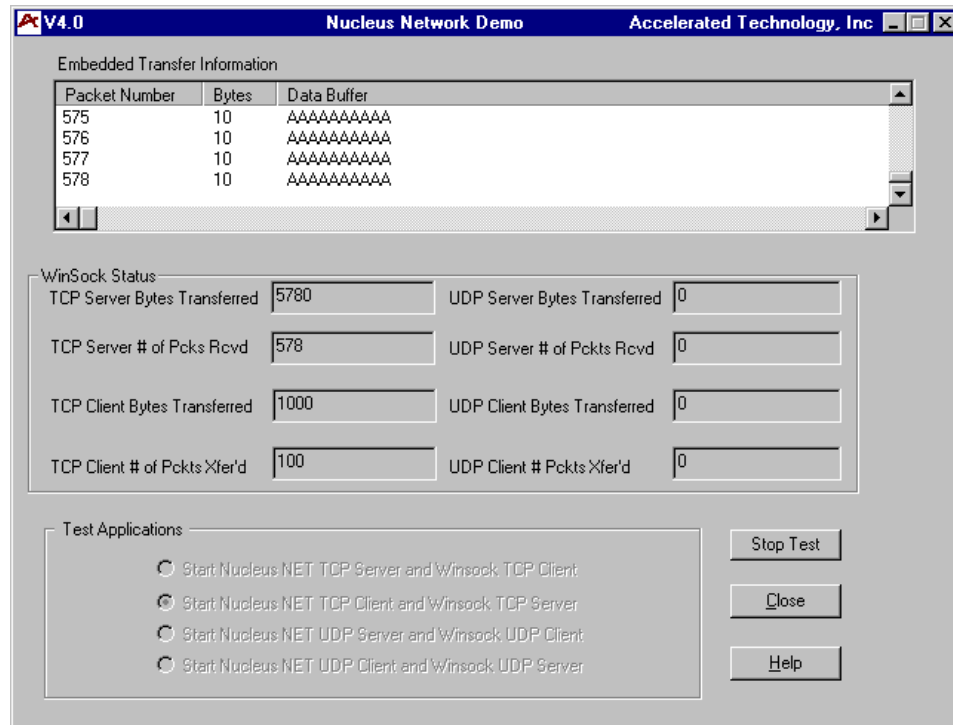


Figure 7.14 Stopping the Test

Exiting the Program

To exit the program, click the *Close* button.





Nucleus PQUICC Driver

Introduction

Building the Nucleus PQUICC Library
from a DOS Prompt

Building and Executing the PQUICC
Driver Demos From a DOS Prompt

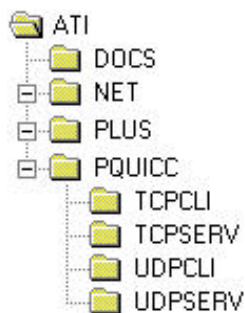
Other Nucleus NET Interface Calls

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

Introduction

This chapter describes the installation of the PQUICC Ethernet Controller driver for Nucleus NET. This driver is designed to work with three boards - the Motorola FADS 821 or 860, with DIAB compiler, and SDS BDM debugger; the Motorola MBX 821 or 860, with DIAB compiler, and SDS BDM debugger; and the Embedded Support Tools (EST) SBC8XX with DIAB compiler, and SDS BDM debugger or EST BDM debugger. PQUICC stands for Motorola Power Quad Integrated Communications Controller. The Serial Communication Controller 1 is configured as an ethernet controller.

To install the Nucleus PQUICC Ethernet Driver software, execute `SETUP.EXE` from the distribution software. All Nucleus products associated with this Motorola PowerPC 821/860 port using the Diab Data Tools must be installed in a common directory (`C:\ATI` by default). PQUICC will require PLUS and NET to be present in this directory structure. PQUICC expects to be within this directory structure and will not build otherwise. After installation, the directory structure should resemble:



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)

Building The Nucleus PQUICC Library From a DOS Prompt

It is recommended that you build the Nucleus PLUS and the Nucleus NET demonstration applications before attempting to build Nucleus PQUICC.

Once the Nucleus products have been installed, and the driver (if any) has been installed, you will be ready to build the Nucleus PQUICC library, `PQUICC.LIB`. Three boards are supported, listed alphabetically, the Embedded Systems Tools (EST) SBX8XX, the Motorola FADS 821/860, and the Motorola MBX 821/860. The MS-DOS batch file contains the Diab Data assembler, compiler, and librarian commands necessary to build the Nucleus PQUICC library. The default compiles for the MPC860 FADS board. To build the PQUICC library, simply enter the following command at an DOS prompt.

The Nucleus library can be built by executing the following command :

```
\ATI\PQUICC> pquicc fads <cr> --Builds for the Motorola FADS board.
```

or

```
\ATI\PQUICC> pquicc est <cr> --Builds for the EST SBC860 board.
```

or

```
\ATI\PQUICC> pquicc mbx <cr> --Builds for the Motorola MBX board.
```

Execution of the file `PQUICC.BAT` will produce the `PQUICC.LIB` library file. It is assumed that interrupts will be used.

Description of Driver Files

The file `PQUICC.C` contains the interface routines to Nucleus NET, a utility routine for initialization, an interrupt service routine, a FIFO routine for transmitting packets, and a routine for receiving packets.

Each device on a network has a unique address called a hardware address. If two devices share a hardware address both devices will confuse each other. PQUICC sets the hardware address within `PQUICC.C`. To ensure your address is unique from anyone else who is running PQUICC, change any one of the following lines in `PQUICC.C` to be any arbitrary values:

```
device->dev_mac_addr[0] = 0x66;
device->dev_mac_addr[1] = 0x55;
device->dev_mac_addr[2] = 0x44;
device->dev_mac_addr[3] = 0x33;
device->dev_mac_addr[4] = 0x14;
device->dev_mac_addr[5] = 0x11;
```



The file `PQUICC.H` contains the necessary symbolic constants and data structure type definitions. The most important data type is the PDA structure. A pointer of this type, called `PQUICC`, is used to overlay the memory area that maps to the registers of the Communication Processor Module (CPM). Once this pointer is initialized to point at the address of the internal RAM area, all operations on the `PQUICC` registers are performed by accessing the appropriate field in this data structure.

Building and Executing the PQUICC Driver Demos From a DOS Prompt

The `PQUICC` demos operate like the `NET` drivers except the `PQUICC` demos use the `PQUICC` driver. See Chapter 7 for information about the `NET` demos. This section contains information about what was changed to make the `NET` demos use the `PQUICC` driver. Each demo is built using the commands listed here:

```
\ATI\PQUICC\TCPCLI> tcpcli <cr>
\ATI\PQUICC\TCPSERV> tcpserv <cr>
\ATI\PQUICC\UDPCLI> udpcli <cr>
\ATI\PQUICC\UDPSERV> udpserv <cr>
```

Each of the demos include a prototype for the `PQUICC` driver initialization function. It is listed below.

```
extern STATUS PQUICC_Init(DV_DEVICE_ENTRY *device);
```

Each driver contains driver is initialized code like the following:

```
devices[0].dv_name = "PQUICC_0";
devices[0].dv_hw.ether.dv_irq = DEMO_IRQ;
devices[0].dv_hw.ether.dv_io_addr = DEMO_IO_ADDR;
devices[0].dv_hw.ether.dv_shared_addr = 0;
devices[0].dv_init = PQUICC_Init;
devices[0].dv_flags = 0;
memcpy (devices[0].dv_ip_addr, serv_ip_addr, 4);
memcpy (devices[0].dv_subnet_mask, subnet, 4);

if (NU_Init_Devices(devices, 1) != NU_SUCCESS)
{
    printf("NU_Init_Devices failed.\n");
    DEMO_Exit(6);
}
```



`dv_name` is a unique name for that device. `dv_init` is set to the PQUICC initialization function. `cd_flags` is set to 0. The **`memcpy`** commands copy the demo's IP address and subnet mask into the device structure. `dv_hw.ether.dv_io_addr`, `dv_hw.ether.dv_irq`, and `dv_hw.ether.dv_shared_addr` are all unused by PQUICC. They are left to set be used for example code if another driver might be used. The last statement initializes the device driver.

Other Nucleus NET Interface Calls

No PQUICC driver routines should be accessed directly by the developer. The PQUICC driver actually provides an interface that is used directly by Nucleus NET without developer intervention.





Nucleus FEC860T Driver

Introduction

Installing the Nucleus FEC860T
Driver Software

Building the Nucleus NET FEC860T
Driver from a DOS Prompt

The FEC860T Driver and
Nucleus NET

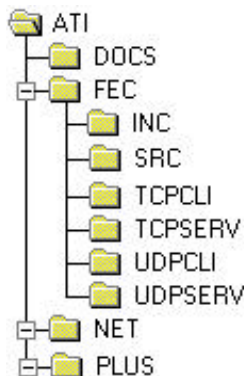
Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.



Introduction

This chapter describes the installation and usage of the MPC860T Fast Ethernet Controller (FEC860T) driver for Nucleus NET. This driver is designed to work with the Motorola FEC860T FADS daughtercard, attached to the MPC8xx FADS board, using the DIAB DATA C compiler, and SDS SingleStep BDM debugger.

To install the Nucleus FEC860T Ethernet Driver software, execute `SETUP.EXE` from the distribution software. All Nucleus products associated with this Motorola PowerPC 821/860 port using the Diab Data Tools must be installed in a common directory (`C:\ATI` by default). The FEC860T Driver will require PLUS and NET to be present in this directory structure. The FEC860T Driver expects to be within this directory structure and will not build otherwise. After installation, the directory structure should resemble:



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)



Building The Nucleus FEC 860T Library From a DOS Prompt

Once the Nucleus products files have been installed, you will be ready to build the Nucleus FEC 860T library, `FEC.LIB`. The Motorola FADS 821/860 is the only board supported with the FEC driver. The MS-DOS batch file contains the Diab Data assembler, compiler, and librarian commands necessary to build the Nucleus FEC library. To build the FEC library, simply enter the following command at an DOS prompt.

```
\ATI\FEC> fec <cr>
```

Execution of the file `FEC.BAT` will produce the `\ATI\FEC\O\FEC.LIB` library file. It is assumed that interrupts will be used.

Description of Driver Files

The file `FEC_EXTR.H` contains the public interface of the FEC860T driver for Nucleus NET. This file must be included by application sources that need to initialize the FEC860T driver. Data types and prototypes of routines used during initialization are found in this file.

The file `FEC.C` contains the implementation of the Nucleus NET driver for the MPC860T Fast Ethernet Controller. This implementation includes initialization, interrupt service, packet transmit, and packet receive routines.

The file `FEC_DEFS.H` contains constants, macros, and type definitions used by `FEC.C` to implement the Nucleus NET driver. This file contains constants and macros detailing the specific hardware interface of the MPC860T Fast Ethernet Controller.

The file `LXT970A.H` contains constants that define the MII registers specific to the LXT970A MII PHY present on the MPC860T FADS daughtercard. These constants are not currently used by the FEC860T driver, but may be of interest to application programmers.

The file `TD_EXTR.H` contains debugging macros used in the FEC860T driver implementation.

The file `TDC.C` contains a utility routine used by the debugging macros present in the file `TD_EXTR.H` and used in the FEC860T driver implementation.



Building and Executing the FEC860T Driver Demos From the DOS Prompt

This section discusses Nucleus NET issues that are specific to the FEC860T driver.

The FEC860T demos operate like the NET drivers except the FEC860T demos use the FEC860T driver. See Chapter 7 for information about the NET demos. This section contains information about what was changed to make the NET demos use the FEC860T driver. Each demo is built using the commands listed here:

```
\ATI\FEC\TCPCLI> tcpcli <cr>
\ATI\FEC\TCPSERV> tcpserv <cr>
\ATI\FEC\UDPCLI> udpcli <cr>
\ATI\FEC\UDPSERV> udpserv <cr>
```

Each of the demos include a prototype for the FEC driver initialization function. It is listed below:

```
extern STATUS FEC_Init(DV_DEVICE_ENTRY *device);
```

Both an array of type `NU_DEVICE` and an `FecParams` structure are needed for FEC860T driver initialization. FEC driver-specific options are placed into the `FecParams` structure via the `fec_SetParams` driver routine. The `NU_DEVICE` array and `FecParams` structure are only needed for the duration of the `NU_Init_Devices` call, and can be discarded afterwards.

The `fec_SetParams` driver routine is used to set the Ethernet hardware MAC address that will be programmed into the MPC860T Fast Ethernet Controller and to supply two optional auto-negotiation parameters to the driver. The supported options, defined in the `FEC_EXTR.H` file, include:

Option	Meaning
<code>FEC_FORCE_10MBPS</code>	Force PHY to advertise 10 Mbps, even if 100 Mbps is available.
<code>FEC_FORCE_HALF_DUPLEX</code>	Force PHY to advertise half-duplex operation, even if full-duplex operation is available.

The `dv_irq` field of the `NU_DEVICE` structure specifies which MPC860T SIU interrupt level is to be used by the Fast Ethernet Controller and the FEC860T driver. The FEC860T driver does not use the `dv_io_addr` or the `dv_shared_addr` fields of the `NU_DEVICE` structure.



Each driver contains driver is initialized code like the following:

```

NU_DEVICE devices[1];
FecParams fecParams;

FEC_SetParams(&fecParams,
              FEC_FORCE_10MBPS | FEC_FORCE_HALF_DUPLEX,
              "\x00\x00\x33\x44\x55\x66");

if (FEC_SetParams(&fecParams, 0, FEC_HW_ADDR) != NU_SUCCESS)
{
    printf("error at call to FEC_SetParams from
TCP_Server_Task.\n");
    DEMO_Exit(12);
}
devices[0].dv_name = "FEC860T_0";
devices[0].dv_hw.ether.dv_irq = DEMO_IRQ;
devices[0].dv_hw.ether.dv_io_addr = DEMO_IO_ADDR;
devices[0].dv_hw.ether.dv_shared_addr = 0;
devices[0].dv_init = FEC_Init;
devices[0].dv_flags = 0;
devices[0].dv_driver_options = (UINT32)&fecParams;
memcpy (devices[0].dv_ip_addr, serv_ip_addr, 4);
memcpy (devices[0].dv_subnet_mask, subnet, 4);

if (NU_Init_Devices(devices, 1) != NU_SUCCESS)
{
    printf("NU_Init_Devices failed in TCP_Server_Task.\n");
    DEMO_Exit(13);
}

```



Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

`dv_name` is a unique name for that device. `dv_init` is set to the FEC initialization function. `cd_flags` is set to 0. The **memcpy** commands copy the demo's IP address and subnet mask into the device structure. The call to `FEC_SetParams` fills the variable `fecParams`. This value is copied into the device structure by setting `dv_driver_options` to the value of `fecParams`. `dv_hw.ether.dv_irq`, `dv_hw.ether.dv_io_addr`, and `dv_hw.ether.dv_shared_addr` are all unused by FEC. These statements are left in this demo to be an example for other drivers that might use them. The last statement initializes the device driver.

Other Nucleus NET Interface Calls

Other than `FEC_SetParams`, which simply builds the FEC-specific initialization parameter block, no FEC860T driver routines should be accessed directly by the developer. The FEC860T driver actually provides an interface that is used directly by Nucleus NET without developer intervention.



10

Nucleus PPP Driver

Introduction

**Building and Executing the PPP
Demonstration System From a
DOS Prompt**

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

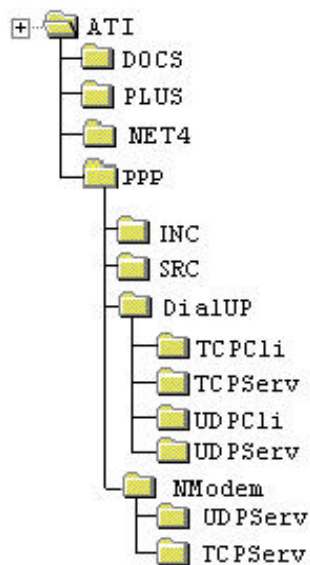


Introduction

The following chapter will discuss the installation of Nucleus PPP, and building the Nucleus PPP library.

Note: Nucleus PLUS and NET must be installed, before Nucleus PPP is installed. It is recommended that you build and execute the Nucleus PLUS and NET demonstration applications before attempting to build Nucleus PPP.

To install the Nucleus PPP software, execute `SETUP.EXE` from the distribution software. The Nucleus PLUS library and all other PLUS files must be in the PLUS directory as shown below. After installation, the directory structure should resemble:



PPP depends on this directory structure and will not build correctly unless the batch files are modified.

Each of the directories, `\TCPCLI`, `\TCPSERV`, `\UDPSERV`, and `\UDPCLI` contain the corresponding demos for testing dialup and nmodem connections. Please see the *Nucleus PPP Driver Reference* manual for more information.



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)

Building the Nucleus PPP Library From a DOS Prompt

Make the following modifications (if applicable):

If PPP will be run over a dialup or null modem connection, edit the `PPP_OPTS.H` file in the PPP directory. Find the line that reads:

```
#define LCP_DEFAULT_AUTH_PROTOCOL
```

and set the value to the PAP value (0xc023).

Edit the `HOSTS.C` file in the NET directory. Add a name and IP address for your networked computer. Use the same format as the example entries that are already in `HOSTS.C`. Be sure the entry you add comes before the NULL entry that terminates the list.

Rebuild the NET library for PPP:

```
\ATI\PPP> netppp <cr>
```

Build the PPP library:

```
\ATI\PPP> ppp <cr>
```



Building and Executing the PPP Demonstration System From a DOS Prompt

In the Nucleus PPP Driver release are six applications used to test the code. The demos reside in \DIALUP and \NMODEM directories.

Each demo may be built by running their respective batch file.

The Null Modem TCPSEV.C and UDPSERV.C demos are intended to be used with a null modem and they demonstrate a tcp server and udp server respectively. Make sure PAP is set in PPP_OPTS.H for dial-up demos and nmodem demos.

The Dialup TCPCLI.C and UDPCLI.C demos are intended to be used with a modem and they demonstrate a dial-up tcp client and udp client respectively. The Server demos were not tested and are included as templates.

Make the following modifications to the demo source files (as applies):

If TCPCLI or UDPCLI will be run over a dialup connection, find the following lines:

```
#define NUMBER_TO_DIAL
```

Set the value to the access number of your PPP server:

```
#define LOGIN_ID
```

Set the value to your login on your PPP server:

```
#define PASSWORD
```

Set the value to your password on your PPP server:

```
char server_name[] = "server_name";
```

Set the value to the machine name that you specified in HOSTS.C.

Instructions for Setting Up TCPserv and UDPServ Demos in \NMODEM

The batch files default to the FADS board configuration. If you have one of the other 821/860 boards, either change the default at the top of the batch files or use one of the following as a command-line parameter while building:

fads - Motorola 821/860 FADS board

est - Motorola 821/860 EST board

mbx - Motorola 821/860 MBX board

Build either Tcpserv or Udpser using the respective batch file:

```
\ATI\PPP\NMODEM\TCPSEV> tcpserv <cr>
```

```
\ATI\PPP\NMODEM\UDPSERV> udpser <cr>
```



An executable .ELF file will be generated in the o\ sub-directory of the current demo directory.

TCPSESV (Nmodem)

The MPC821/860 board should be connected to an available COM port on your PC as follows:

- FADS -- straight-through serial cable connected to the bottom DB9 connector on the FADS board.
- EST -- straight-through serial cable with DTR and DSR tied together connected to the JP3 connector on the EST board. You may need an adaptor that ties DTR and DSR together.
- MBX -- null-modem cable connected to the DB9 connector on the MBX board.

Tcpserv for null-modem is a program that executes on the MPC821/860 board and services requests from a TCP client program running on a Windows 95/98/NT PC.

In Windows, double-click the "My Computer" icon, then choose "Dial Up Networking". Execute the wizard, and create a dial-up connection that connects to a standard 28800 modem. Refer to the Windows Help for details on installing the dial-up connection.

After the dial-up connection has been created, edit its properties. Click the "Configure" button below the modem selection and set the maximum speed to 115200. Click the "Connection" tab and set the connection preferences to 8-N-1. Click the "Advanced" button and uncheck "Use flow control". Next, go to "My Computer/Control Panel/Modem" and verify that all the settings match with those you set in dial-up networking.

Next, run the TCPSESV Demo in the SDS onchip debugger. Choose *File* and then select *Debug* to upload the o\TCPSESV.ELF executable to the target. Select the *Processor* tab and select "By Object File". Select the *Options* tab and uncheck "Require Exact Symbol Names". The rest of the options are checked. Click *OK* to upload code. Click the "green light" icon (or "GO" on the menu) to begin execution of the demo.

Start the dial-up connection you created earlier and click on the 'Dial' button. Enter the username of "joe", and the password of "blow". Phone number is optional. Then click the 'OK' button. You will see some status messages, then you should see the connection established. At this point, Windows will think it is connected to a network over a PPP link. If you are using Windows 95/98, you may need to disable your Ethernet connection so that Windows will send any network traffic to the PPP dial-up connection. In Windows NT, the routing will be setup automatically, so the Ethernet connection can remain enabled.



Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

The NetDemo application, found in the NET\NETDEMO directory, can now be used to communicate with the TCP server running on the MPC821/860 board. Click *"Start Nucleus NET TCP Server and Winsock TCP Client"* as the option. The IP address of the TCP server running on the MPC821/860 board is 192.200.100.50, and the port number is 7.

UDPSERV (Nmodem)

The UDPSERV program for null-modem is executed as described above for TCPSERV.

Instructions for Setting Up TCPcli and UDPcli Demos in \DIALUP

The batch files default to the FADS board configuration. If you have one of the other 821/860 boards, either change the default at the top of the batch files or use one of the following as a command-line parameter while building:

fads - Motorola 821/860 FADS board

est - Motorola 821/860 EST board

mbx - Motorola 821/860 MBX board

Build the PPP Demo programs by executing the corresponding batch files.

```
\ATI\PPP\DIALUP\TCPSERV> tcpserv <cr>
\ATI\PPP\DIALUP\TCPCLI> tcpcli <cr>
\ATI\PPP\DIALUP\UDPSERV> udpserve <cr>
\ATI\PPP\DIALUP\UDPCLI> udpcli <cr>
```

An executable .ELF file will be generated in the O\ sub-directory of the current demo directory.

TCPCLI (DialUp)

The MPC821/860 board should be connected to a modem as follows:

- FADS -- null-modem cable with DSR and DTR tied together connected to the bottom DB9 connector on the FADS board. You may need an adaptor that ties DTR and DSR together.
- EST -- straight-through serial cable with DTR and DSR tied together connected to the JP3 connector on the EST board. You may need an adaptor that ties DTR and DSR together.
- MBX -- straight-through cable connected to the DB9 connector on the MBX board. Connect a modem (RJ-11) cable from the modem to an analog phone line.

TCPCLI is a program that executes on the MPC821/860 board and connects to a network using PPP through an external modem. It then sends test strings to a TCP server program running on a Windows 95/98/NT computer on the same network.



Before downloading the client program, run the NetDemo application. Click *"Start Nucleus NET TCP Client and Winsock TCP Server"* as the option. It will initialize some parameters, then go into listen mode.

Next, run the TCPCLI Demo in the SDS onchip debugger. Choose *File* and then select *Debug* to upload the `O\TCPCLI.ELF` executable to the target. Select the *Processor* tab and select *"By Object File"*. Select the *Options* tab and uncheck *"Require Exact Symbol Names"*. The rest of the options are checked. Click *OK* to upload code. Click the *"green light"* icon (or *"GO"* on the menu) to begin execution of the demo.

The TCP Client demo will connect to the PPP server. At this point, the TCP server in NetDemo will receive the test strings being sent by the TCPCLI program running on the MPC821/860 board.

UDPCLI (DialUp)

The UDPCLI program is executed as described above for TCPCLI.

TCPSESV (DialUp)

TCPSESV is a program that executes on the 821/860 board and services requests from a TCP client program running on a Windows 95/98/NT PC that has connected to the server over a telephone line. Both the 821/860 board and the PC should be attached to external modems that connect to the same telephone line.

NOTE: TCPSESV and UDPSERV are templates only and were not tested. If need be, they can be tested with two modems; one for the server and one for a Windows 95 client to connect to.



Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

UDPSERV (DialUp)

The UDPSERV program is built and executed as described above for TCPSERV.

NOTE: TCPSERV and UDPSERV are templates only and were not tested. If need be, they can be tested with two modems; one for the server and one for a Windows 95 client to connect to.



Nucleus FTP Server

Introduction

**Building the Nucleus FTP Library
From a DOS Prompt**

**Building and Executing the FTP
Server Application From a DOS
Prompt**

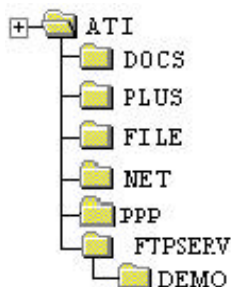
Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

Introduction

This chapter will discuss the installation of the Nucleus FTP Server source code, building the Nucleus FTP Server library, and building the demonstration application.

Note: Nucleus PLUS, NET, a driver, and a file system must be installed, before Nucleus FTP Server is built. It is recommended that you build and execute these components before attempting to build Nucleus FTP Server. For more information concerning Nucleus PLUS refer to the chapters '*Nucleus PLUS*' and '*Nucleus PLUS Usage*' of this manual. For a discussion of the different drivers that can be used with Nucleus NET, see chapters '*Nucleus PQUICC Driver*' and '*Nucleus FEC860T Driver*' of this manual.

To install the Nucleus FTP Server software, execute `SETUP.EXE` from the distribution software. All Nucleus products associated with this Motorola PowerPC 821/860 port using the Diab Data Tools must be installed in a common directory (`C:\ATI` by default). FTP Server will require PLUS and NET to be present in this directory structure. If PQUICC and Nucleus FILE are not present, then the demo will need to be modified to use the substitute driver. FTP Server expects to be within this directory structure and will not build otherwise. After installation, the directory structure should resemble:



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)



Building the Nucleus FTP Server Library From a DOS Prompt

First, make sure the NET library in the PPP directory has been built. Next, make sure the FILE library has been built with no drivers. This can be checked in `\FILE\PCDISK.H`. The constants `RAMDISK`, `RAMDISK_FROMPOOL`, `EBS_FLOPPY`, and `EBS_IDE` should all be set to zero. These constants can be found in the 'ATI Drivers' section. You must then build the FAL library by executing the following command :

```
\ATI\FAL> fal <cr>
```

To build the library, execute the following command:

```
\ATI\FTPSERV> ftpserv <cr>
```

This will compile all the files of the FTP Server library and put all of them into the library file, `\ATI\FTPSERV\O\FTPSERV.LIB`.

Building and Executing the FTP Server Demonstration System From a DOS Prompt

`FTPSDEMO.C` is the demonstration program source code. The program uses a ethernet connection to show up on a network and act as a server for FTP clients. The following line of code found in `FTPSDEMO.C` assigns the server an IP address. This can be changed if your network requires it.

```
#define DEMO_SERVER_IP_ADDR {100, 200, 200, 1};
```

To create the demonstration program type:

```
\ATI\FTPSERV\DEMO> ftpsdemo <cr>
```

Load `FTPSDEMO.ELF` to the PowerPC using Single Step Debugger. The program is loaded just like the PLUS demonstration program was loaded. Run the demo and initiate ftp by opening a DOS box and then by typing the following:

```
>ftp 192.200.100.1
ftp>mkdir test
ftp>dir
    (verify the disk is empty)
ftp>put c:\autoexec.bat
ftp>dir
    (verify the size of the file on the server matches the size on your PC's hard disk)
```



Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

```
ftp>append c:\autoexec.bat autoexec.bat
ftp>dir
(verify the sizeof he file has doubled)

ftp>get autoexec.bat c:\temp\autoexec.bat
(verify your PC has another autoexec.bat in \temp that is twice as large as its
original)

ftp>cd test
ftp>mkdir pub
ftp>dir
(verify that you are in test and there is a pub directory)

ftp>rmdir pub
ftp>dir
(verify pub is gone)

ftp>quit
```



Nucleus FTP Client

Introduction

**Building the Nucleus FTP Client
Library From a DOS Prompt**

**Building and Executing the
Nucleus FTP Client
Demonstration System from a
DOS Prompt**

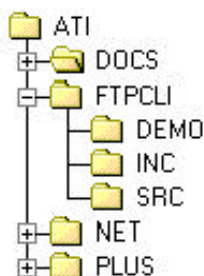
Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

Introduction

This chapter will discuss the installation of the Nucleus FTP Client source code, building the Nucleus FTP Client library, and building the demonstration application.

Note: Nucleus PLUS, NET, and a driver must be installed before Nucleus FTP Client is built. It is recommended that you build and execute the Nucleus PLUS and NET demonstration applications before attempting to build Nucleus FTP Client. For more information concerning Nucleus PLUS refer to the chapters '*Nucleus PLUS*' and '*Nucleus PLUS Usage*' of this manual. For a discussion of the different drivers that can be used with Nucleus NET, see chapters '*Nucleus PQUICC Driver*' and '*Nucleus FEC860T Driver*' of this manual.

To install the Nucleus FTP Client software, execute `SETUP.EXE` from the distribution software. All Nucleus products associated with this Motorola PowerPC 821/860 port using the Diab Data Tools must be installed in a common directory (`C:\ATI` by default). FTP Client will require PLUS and NET to be present in this directory structure. If PQUICC is not present, then the demo will need to be modified to use the substitute driver. FTP Client expects to be within this directory structure and will not build otherwise. After installation, the directory structure should resemble:



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)



Building the Nucleus FTP Client Library From a DOS Prompt

To build the library, execute the following command:

```
\ATI\FTPCLI> ftpcli <cr>
```

This will compile all the files of the FTP Client library and put all of them into the library file, \ATI\FTPCLI\O\FTPCLI.LIB.

Building and Executing the Nucleus FTP Client Demonstration System from a DOS Prompt

FTPDEMO.C is the FTP Client demonstration program. It will connect to an FTP Server program, download a file and upload the contents under a different name.

At the top of the source code in FTPDEMO.C are pre-compiler symbols that should be modified to your network environment.

```
#define printf          PRINTF
#define DEMO_IRQ        5
#define DEMO_IO_ADDR    0x0300L
#define DEMO_CLIENT_IP_ADDR {100,200,300,400}
#define DEMO_SERVER_IP_ADDR {100,200,300,400}
#define DEMO_SERVER_PORT 0
#define USERNAME        "user id"
#define PASSWORD        "password"
#define FILE_TO_GET     "/directory/oldfile.txt"
#define FILE_TO_PUT     "/directory/newfile.txt"
CHAR    pause = 0;
```

The symbol `printf` is stubbed out. It can be set to another function for debugging purposes. `DEMO_IRQ` and `DEMO_ADDR` are not used by the PQUICC driver and are left to be used as an example if another driver is used. `DEMO_CLIENT_IP_ADDR` is the address of the board. `DEMO_SERVER_IP_ADDR` is the address of the machine running the FTP Server the demo will log into. `DEMO_SERVER_PORT` is the port number this FTP Server. `USERNAME` and `PASSWORD` is what the demo will use to log into the FTP Server. Once logged in the demo will request the file that is set to be the symbol `FILE_TO_GET`. It will then upload the contents of that file as `FILE_TO_PUT`. The variable `pause` is a time length to pause between various commands during the demo. This will allow the FTP Server time to react. If this value is 0 the demo will not pause.



Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

To build the FTP Client demonstration program, `FTPDEMO.ELF`, execute the following command:

```
ATI\FTPCLI\DEMO> ftpdemo <cr>
```

FTP Client will work with any FTP Server program. Shareware FTP Server programs can be found on the Internet. The FTP Server program you choose must have an account with that can use the `USERNAME` and `PASSWORD` specified in `FTPDEMO.C`. The FTP Server will ask for a home directory for this account. The `FILE_TO_GET` must reside in this directory. It is best if this is a small text file. `FILE_TO_PUT` should be a file that does not exist in that same home directory. FTP is case sensitive. List the files from a DOS box to be sure your program uses the correct case! An example would be an account “*MyAccount*” that has the home directory `C:\`. The `FILE_TO_GET` would be set to the small file “*autoexec.bat*”. `FILE_TO_PUT` would be “*autoexec.new*” which would not exist in `C:\` yet.

After `FTPDEMO.ELF` has been built it can be downloaded to any of the three supported 821/860 evaluation boards just like the `PLUS` demo is loaded. Remember to start the FTP Server application before starting the embedded FTP Client demo. Depending on what FTP Server program you are using there should be output verifying that the FTP Client is executing valid commands. After a short while the file specified by `FILE_TO_PUT` should be created in the home directory. It should look identical to the file specified by `FILE_TO_GET`.



Nucleus TFTP Server

Introduction

**Building and Executing the TFTP
Demo Application From a DOS
Prompt**

**Building and Executing the TFTP
Demonstration System From a
DOS Prompt**

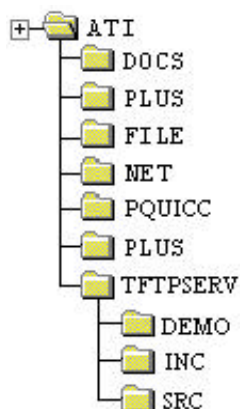
Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

Introduction

The following chapter will discuss the installation of the Nucleus TFTP Server, and building the Nucleus TFTP library.

Note: Nucleus PLUS, NET, and a driver must be installed, before Nucleus TFTP is installed. It is recommended that you build and execute the Nucleus PLUS and NET demonstration applications before attempting to build Nucleus TFTP.

To install the Nucleus TFTP software, execute `SETUP.EXE` from the distribution software. The Nucleus PLUS library and all other PLUS files must be in the PLUS directory as shown below. After installation, the directory structure should resemble:



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step 7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)



Building the Nucleus TFTP Library From a DOS Prompt

First, make sure the NET library in the network driver library has been built.

To build the TFTP library, the user should execute the following command:

```
\ATI\TFTPSERV> tftpserv <cr>
```

This will compile all the files of the TFTP library and put all of them into the library file, `TFTPSERV.LIB`.

Building and Executing the TFTP Demonstration System From a DOS Prompt

Building the Demo

The TFTP demo application must be configured for use on your system. There is an IP address that must be changed. In `TFTPSDEM.C` search for the following line:

```
#define DEMO_SERVER_IP_ADDR {192,39,168,25}
```

This is the IP address of the TFTP server application. Modify this address for use on your network.

To compile and link the TFTP demonstration program, execute the following command:

```
\ATI\TFTPSERV\DEMO> tftpsdem <cr>
```

Executing the Demo

Load `TFTPSDEM.ELF` to the PowerPC using Single Step Debugger. The program is loaded just like the PLUS demonstration program was loaded. Run the demo and initiate TFTP by opening a DOS box and then typing the following on an NT System, Windows 95 will require a TFTP Client test program:

```
>tftp 192.200.100.25 put test.txt test.txt
Transfer successful: 749 bytes in 1 second, 749 bytes/s
```

Any text file will do for the test, just replace a `TEST.TXT` in the command above with a text file of your choice.





Nucleus TFTP Client

Introduction

**Building the TFTP Library From a
DOS Prompt**

**Building and Executing the TFTP
Demo Application From a DOS
Prompt**

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

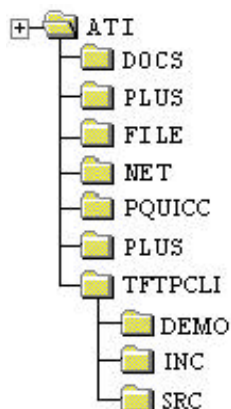


Introduction

The following chapter will discuss the installation of the Nucleus TFTP Client, and building the Nucleus TFTP library.

Note: Nucleus PLUS, NET, and a driver must be installed, before Nucleus TFTP is installed. It is recommended that you build and execute the Nucleus PLUS and NET demonstration applications before attempting to build Nucleus TFTP.

To install the Nucleus TFTP software, execute `SETUP.EXE` from the distribution software. The Nucleus PLUS library and all other PLUS files must be in the PLUS directory as shown below. After installation, the directory structure should resemble:



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step 7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)



Building the Nucleus TFTP Library From a DOS Prompt

The batch file `TFTPCLI.BAT` is provided for building the TFTP library and the batch file `TFTPDEMO.BAT` builds the demo program.

To build the TFTP library, execute the following command:

```
\ATI\TFTPCLI> tftpcli <cr>
```

This will compile all the files of the TFTP library and put all of them into the library file, `TFTPCLI.LIB`.

Building and Executing the TFTPDEMO Application From a DOS Prompt

The `TFTPDEMO.BAT` builds the TFTP source files and TFTP demo application and links it all together to produce `TFTPDEMO.ELF`, using the following command prompt:

```
\ATI\TFTPCLI\DEMO> tftpdemo <cr>
```

The `TFTPDEMO.ELF` file should be downloaded to a evaluation board using SingleStep BDM debugger. The TFTP demo application must be configured for use on your system. There are two IP addresses that must be changed. In `TFTPDEMO.C` search for the following two lines:

```
#define DEMO_SERVER_IP_ADDR {192,39,168,215}
#define DEMO_CLIENT_IP_ADDR {192,39,168,25}
```

The first IP address is the address of a TFTP server on your network. The second is the IP address of the TFTP client application. Modify these addresses for use on your network.

When executed the demo first tries to establish a connection with the TFTP server. Once the connection is established the client attempts to put a file named `TEST.TXT` to the server.

If the file is successfully sent to the server, the TFTP demo will then attempt to get it back. The TFTP demo then performs a comparison of the file sent with the one retrieved to make sure that they are identical.





Nucleus SNMP

Introduction

**Rebuilding Other Nucleus
Libraries**

**Building the Nucleus SNMP
Library from a DOS Prompt**

**Building and Running the
SNMP Demonstration System
From a DOS Prompt**

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

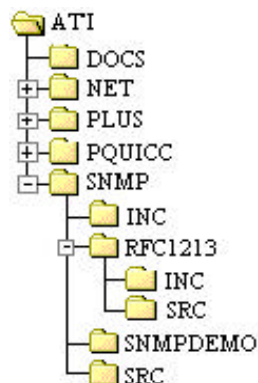


Introduction

This chapter will discuss the installation of the Nucleus SNMP source code, building the Nucleus SNMP library, and building the demonstration application.

Note: Nucleus PLUS, NET, and a driver must be installed before Nucleus SNMP is built. It is recommended that you build and execute the Nucleus PLUS and NET demonstration applications before attempting to build Nucleus SNMP. For more information concerning Nucleus PLUS refer to the chapters '*Nucleus PLUS*' and '*Nucleus PLUS Usage*' of this manual. For a discussion of the different drivers that can be used with Nucleus NET, see chapters '*Nucleus PQUICC Driver*' and '*Nucleus FEC860T Driver*' of this manual.

To install the Nucleus SNMP software, execute `SETUP.EXE` from the distribution software. All Nucleus products associated with this Motorola PowerPC 821/860 port using the Diab Data Tools must be installed in a common directory (`C:\ATI` by default). SNMP will require PLUS and NET to be present in this directory structure. If PQUICC is not present, then the demo will need to be modified to use the substitute driver. SNMP expects to be within this directory structure and will not build otherwise. After installation, the directory structure should resemble:



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)



Rebuilding Other Nucleus Libraries

SNMP manages the statistics that are logged from other parts of the network stack. Nucleus NET and the driver must be built differently so these libraries will report this information to Nucleus SNMP. `NET_S.LIB`, `PQUICC_S.LIB` and `FEC_S.LIB` are the NET, PQUICC, and FEC libraries built to log SNMP statistics. Each is kept in the directory with the original library. A library built to log SNMP statistics must be linked with the SNMP library. Hence, when verifying the driver demos or a Nucleus NET product that does not use SNMP they will need to be linked with the usual libraries (`NET.LIB`, `PQUICC.LIB` and `FEC.LIB`). However, the SNMP demo application looks for the SNMP variations of the libraries it needs. `NET_S.BAT`, `PQUICC_S.BAT` and `FEC_BAT` will build the corresponding SNMP enabled libraries. These can be run by hand if needed. It may be easier to run `SNMP.BAT`. `SNMP.BAT` will build `NET_S.LIB` and `PQUICC_S.LIB` if needed. `SNMP.BAT` will not build `FEC_S.LIB` because the SNMP demonstration system is shipped to run with the PQUICC driver. To test SNMP with the FEC driver, modify the `SNMP.BAT` or run `FEC_S.BAT` manually.

Building the SNMP Library From a DOS Prompt

Before building the SNMP library, you must edit the file `SNMP_CFG.C`. `SNMP_CFG.C` is located in `\ATI\SNMP\SRC`. There is an array of IP addresses that control what addresses the SNMP manager will respond to. Do not change the size of this array. Replace the placeholder entries with the IP address of the machines you want to use to query the embedded board. For example, change "Host1" to a label that is meaningful for your environment. Keep the field "public". Change the hex number `0xC0C86432UL` to the hex equivalent of the IP address for that machine.

The batch file `SNMP.BAT` will build the Nucleus SNMP library.

```
\ATI\SNMP> snmp <cr>
```

This will create `\SNMP\O\SNMP.LIB` will be created.

Building and Executing the SNMP Demonstration System from a DOS Prompt

The demonstration application is simply an TCP Server demonstration that has an SNMP task running in the background. Hence, the SNMP demo (`SNMPAPP.C`) code works the same way the NET TCP Server demo (`TCPSEV.C`) works. `SNMPAPP.C` accepts a limited number of connections from clients and echos the data the client sends. Before building the demo modify the following lines of code:

```
#define DEMO_IRQ                5
#define DEMO_IO_ADDR            0x0300L
#define DEMO_MAX_CONNECTIONS    10
#define DEMO_SERVER_IP_ADDR     {100,200,300,400}
```



Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

DEMO_IRQ and DEMO_IO_ADDR are driver specific defines that are not used in this compile only demo. DEMO_MAX_CONNECTIONS is the number of pending connections the server will accept. DEMO_SERVER_IP_ADDR is the IP address of the board.

The batch file SNMPAPP.BAT is provided to build the SNMP demo application. To build the demo application execute the following command:

```
ATI\SNMP\SNMPDEMO> snmpapp <cr>
```

The result will be an ELF format executable, \ATI\SNMP\SNMPDEMO\O\SNMPAPP.ELF. SNMPAPP.ELF can be loaded with the SDS SingleStep BDM 8xx debugger. Start the debugger and select SNMPAPP.ELF. Start the SNMPAPP.ELF on the embedded board. Once the application is executing you can try to ping it from a machine on the network. If the response to the ping is received then you can connect from your SNMP manager.



16

Nucleus SMTP

Introduction

**Building the Nucleus SMTP
Library from a DOS Prompt**

**Building and Executing the
Nucleus SMTP Server and Client
Demonstrations from a DOS
Prompt**

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

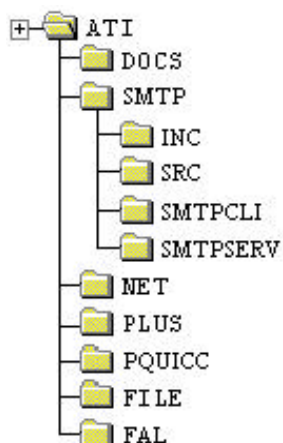


Introduction

The following chapter will discuss the installation of Nucleus SMTP, and building the Nucleus SMTP library.

Note: Nucleus PLUS, NET, and a driver must be installed, before Nucleus SMTP is installed. It is recommended that you build and execute the Nucleus PLUS and NET demonstration applications before attempting to build Nucleus SMTP.

To install the Nucleus SMTP software, execute `SETUP.EXE` from the distribution software. The Nucleus PLUS library and all other PLUS files must be in the PLUS directory as shown below. After installation, the directory structure should resemble:



SMTP depends on this directory structure and will not build correctly if modified, unless the batch files are also modified to reflect the changes.

Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)



Building the Nucleus SMTP Library From a DOS Prompt

Before building the SMTP library, make sure the Nucleus Plus, NET, FAL, and your network driver libraries have been built and tested. **Note:** Be Sure that the file `IFILE.H`, located in `FAL\INC\` defines `INCLUDE_FILE_LAYER`.

To build the SMTP library, execute:

```
\ATI\SMTP> smtp <cr>
```

This will create `O\SMTP.LIB`. It will also build for a specific board in this case the FADS evaluation board.

Consult the *SMTP Reference Manual*, located in `\DOCS`, for more information about SMTP.

Building and Executing the Nucleus SMTP Client Demonstration from a DOS Prompt

To build the demo, make sure that the SMTP library has been built properly and then execute the following:

```
\ATI\SMTP\SMTPCLI> smtpcli <cr>
```

This will build and link all files necessary to run the Nucleus SMTP client demonstration.

The SMTP client demo application must be configured for use on your system. There are two IP addresses that must be changed. In `SMTPCLI.C` search for the following two lines:

```
#define DEMO_SERVER_IP_ADDR {192,39,168,215}
#define DEMO_CLIENT_IP_ADDR {192,39,168,25}
```

The first IP address is the address of a SMTP server on your network. The second is the IP address of the SMTP client application. Modify these addresses for use on your network.



Nucleus Target Specific Notes PowerPC 821/860 Diab Data Tools

The SMTP client was tested using the shareware email program IMail. If you do not have access to this application, just setup another email program in a similar manner. An account is setup according to the parameters in the client demo, and then the demo is run. The demo program sends a mail message to several accounts set up in the Imail application. Follow the instructions below to setup and test the SMTP client demo:

- Install IMAIL and open the Imail Administrator from the *Start* menu.
- Create the virtual domain name hgoober.com.
- Add the following users: joeblow, fishman, and don.
- Download and run the SMTPCLI to the embedded board and run.
- After about 30 seconds, close the Imail Administrator and open the Imail Client.
- Log on as each of the 3 users and check the mailbox for the email.
- Each user should receive the same email.
- Open up telnet and make a connection to the IP address of your Nucleus SMTP Server(192.239.168.25) and set the Port to 25. This test will allow you test the help functions, helo command and ehlo command. Once connected, type: **help**. This will type all help commands.

You do not have to worry about changing the passwords because SMTP will be used to send the file to the IMAIL's SMTP server.

Building and Executing the Nucleus SMTP Server Demonstration from a DOS Prompt

To build the demo, make sure that the SMTP library was built properly and then execute the following:

```
\ATI\SMTP\SMTPSERV> smtpserv <cr>
```

This will build and link all files necessary to run the Nucleus SMTP demonstration. Download and execute in the same manner as you did the PLUS demo.



The SMTP Server demo application must be configured for use on your system. There are two IP addresses that must be changed. In `SMTPSERV.C` search for the following two lines:

```
#define DEMO_SERVER_IP_ADDR {192,39,168,215}
#define DEMO_CLIENT_IP_ADDR {192,39,168,25}
```

The first IP address is the address of a SMTP server on your network. The second is the IP address of the SMTP client application. Modify these addresses for use on your network.

The SMTP server demonstration is tested using *Outlook Express* or similar email client.

- In *Outlook*, go to the *tools* menu item and select the *accounts* sub-menu item. Then click *add* and select mail account.
- Type *john doe* as the name. Click *Next*.
- Type *john@nuc.smtp.net* for the email address. Click *Next*
- Fill in the test IP address for the POP3/IMAP server, which is the IP address of the embedded board. Click *Next*.
- For POP account name enter *john* and the password *John*. This is an SMTP demo so the POP server will be accessed by *Outlook*, but we do not have a POP server in this demo. Click *Next*.
- Select the default for the internet account name and click *Next*.
- Select the connection type and use my *local area network* to access the Internet. Click *Next*. Click *Finish*.
- Now select the new account IP address as default. Click *Close*.

Outlook is now set up so click *Compose* and create a message to *jane@nuc.smtp.net*. Send the message and verify that it was sent by clicking on the *SEND/RECEIVE* button and click the details. You should see the message being sent. Now mail a message to *willy@nuc.smtp.net*, and *john@nuc.smtp.net*. If all messages are sent, then the SMTP server demo is working correctly.





Nucleus POP3

Introduction

**Building the Nucleus POP3
Library from a DOS Prompt**

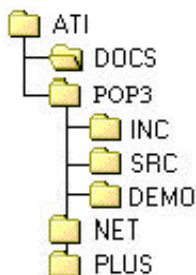
**Building and Executing the
Nucleus POP3 Demonstrations
from a DOS Prompt**

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

Introduction

The following chapter will discuss the installation of Nucleus POP3, and building the Nucleus POP3 library. Note: Nucleus PLUS, NET, and a driver must be installed, before Nucleus POP3 is installed. It is recommended that you build and execute the Nucleus PLUS and NET demonstration applications before attempting to build Nucleus POP3.

To install the Nucleus POP3 software, execute `SETUP.EXE` from the distribution software. The Nucleus PLUS library and all other PLUS files must be in the PLUS directory as shown below. After installation, the directory structure should resemble:



Tools and Hardware

This product was compiled with the following tools:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)

Building the Nucleus POP3 Library From a DOS Prompt

The POP3 library and demonstration programs were tested using a MPC860 FADS board. The PQUICC ethernet driver was used for network connectivity. Nucleus FAL is also required.

Note: Be sure that the file `IFILE.H`, located in `FAL\INC\` defines `NUCLEUS_FILE_INCLUDED`.



The following build process should be used:

- Build Nucleus Plus, NET, FAL, and the network driver as instructed in their respective chapters.
- POP3 was tested using the PQUICC ethernet driver. If you plan on using your own driver, build and link it instead of PQUICC.
- Build the POP3 library. This is done by executing:

```
\ATI\POP3> pop3 FADS <cr>
```

This will create O\POP3.LIB. It will also build for a specific board in this case the FADS evaluation board.

Consult the POP3 Reference Manual, located in \DOCS, for more information about POP3.

Building and Executing the POP3 Demonstration System From a DOS Prompt

Before building the POP3 Demo, be sure that the POP3 library is built. There are four parameters that need to be setup prior to building the demo. They are as follows:

Parameter	Description
DEMO_CLIENT_IP_ADDR	The IP address of the device that you are to run POP3 on.
DEMO_SERVER_IP_ADDR	The IP address of the POP3 server that you are to communicate with.
ps.user	The user ID for an account on the POP3 server.
ps.pass	The associated password of the account on the POP3 server.

To build the POP3DEMO application, execute the following command:

```
\ATI\POP3\DEMO> pop3demo <cr>
```

This will build and link all files necessary to run the Nucleus POP3 demonstration.

Before testing the demo, it is a good idea to populate the user account that will be used for the test with 5 test emails.

Download the demo in the same manner as you did the PLUS demo. The best way to test if it worked for an embedded user is to set breakpoints in the POP3DEMO.C file at locations where each test might fail, and set one for when it has completed at the end of the tests. If the program reaches the end of the task before hitting any of the error conditions, it is functioning properly.

As an added test, you can look at the user's record on the SMTP server that POP3DEMO is connecting to so that you may verify the account was accessed. This is not necessary if the POP3DEMO encounters no errors since it will have run properly.





Nucleus Telnet

Introduction

**Building the Nucleus Telnet
Library From a DOS Prompt**

**Building the Nucleus Telnet
Demonstration System From
a DOS Prompt**

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

Introduction

This chapter will discuss the installation of the Nucleus Telnet Server source code, building the Nucleus Telnet library, and building the demonstration application.

Note: Nucleus PLUS, NET, and a driver must be installed, before Nucleus Telnet is built. It is recommended that you build and execute the Nucleus PLUS and NET demonstration applications before attempting to build Nucleus Telnet. For more information concerning Nucleus PLUS refer to the chapters '*Nucleus PLUS*' and '*Nucleus PLUS Usage*' of this manual. For a discussion of the different drivers that can be used with Nucleus NET, see chapters '*Nucleus PQUICC Driver*' and '*Nucleus FEC860T Driver*' of this manual.

To install the Nucleus Telnet software, execute `SETUP.EXE` from the distribution software. All Nucleus products associated with this Motorola PowerPC 821/860 port using the Diab Data Tools must be installed in a common directory (`C:\ATI` by default). Telnet will require PLUS and NET to be present in this directory structure. If PQUICC is not present, then the demo will need to be modified to use the substitute driver. Telnet expects to be within this directory structure and will not build otherwise. After installation, the directory structure should resemble:



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)



Building the Nucleus Telnet Library From a DOS Prompt

To build the library, execute the following command:

```
\ATI\TELNET> telnet <cr>
```

This will compile all the files of the TELNET library and put all of them into the library file, \ATI\TELNET\O\TELNET.LIB.

Building and Executing the Telnet Demonstration System From a DOS Prompt

TN_DEBUG.C is Nucleus Debug+ that interfaces a telnet client. The demo is configured to use a PQUICC Ethernet driver. The demo will need to be modified to use another driver. If you have purchased another driver from Accelerated Technology, see the corresponding chapter for information about how to modify the demo to use that driver. Regardless of what driver is used, modify the following lines of code before building the demo:

```
#define printf          PRINTF
#define DEMO_IRQ        5
#define DEMO_IO_ADDR    0x0300L;
#define DEMO_SERVER_IP_ADDR {100,200,300,400}
```

The symbol printf is defined to a function that has been stubbed out. This can be redefined as a function to help debug this demo. DEMO_IRQ and DEMO_IO_ADDR are not used by the PQUICC driver. They have been left to help users who use other drivers that may need that information. DEMO_SERVER_IP_ADDR is the IP address of the board.

To build the TELNET demo execute the following command:

```
\ATI\TELNET\DEMO> tn_debug <cr>
```

This will create the file TN_DEBUG.ELF.

Download this file to the board using the same process described for the PLUS port. For more information see Chapter 2. Once the program has been started on the board, use a telnet client application to communicate with the demonstration application. Information should be displayed to the terminal screen. At this point commands may be entered to the demo running on the board. "Help" lists the available commands.





Nucleus RIP2

Introduction

Building the Nucleus RIP2 Library From a DOS Prompt

Building and Executing the RIP2 Demonstration System From a DOS Prompt

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.

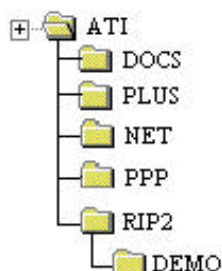


Introduction

This chapter will discuss the installation of Nucleus RIP2, and building the Nucleus RIP2 library.

Note: Nucleus PLUS, NET, and a driver must be installed, before Nucleus RIP2 is installed. It is recommended that you build and execute the Nucleus PLUS and NET demonstration applications before attempting to build Nucleus RIP2.

To install the Nucleus RIP2 software, execute `SETUP.EXE` from the distribution software. The Nucleus PLUS library and all other PLUS files must be in the PLUS directory as shown below. After installation, the directory structure should resemble:



RIP2 depends on this directory structure and will not build correctly unless the batch files are modified.

Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)



Building the Nucleus RIP2 Library From a DOS Prompt

First Nucleus NET must be re-build with RIP2 enabled. Type: `ATI\RIP2 >?bldnet` (where ? = a, m, or e, for the FADS, MBX, or EST board)

Building and Executing the RIP2 Demonstration System From a DOS Prompt

To build the RIP2 demonstration make sure that all paths are set up to the NET and PLUS directories. Then type:

```
\ATI\RIP2\DEMO> ?bldap rip2exec <cr> (where ? = a, m, or e, see above)
```

This will build and link all files necessary to run the Nucleus RIP2 demonstration.

To test the demo the IP address in the demo needs to be setup before running the demo. The variable `ip_addr` is the IP address of the MPC821/860, and must be on the same network as the router to which it is physically connected.

The demo running on the MPC821/860 will periodically output routing tables through the SMC1 serial port. Please refer to the Nucleus PLUS chapter for information on hooking up the MPC821/860 serial port to your computer and seeing output. By default, the baud rate for `RIP2.EXE` is 57600.

Start the demo.

The output will be printed to a terminal program. This output can be stopped by changing the 1 to a 0 in the `RIP2.C` file for the following

```
#defines:
#define HEX_DUMP      1
#define DUMP25        1
#define OUTPUT        1
```

The Nucleus NET library and the demo must then be rebuilt. Verify that the routes corresponds to the router/hub board setup. Tables will be output every few seconds.





20

Nucleus WebServ

Introduction

**Building the Nucleus WebServ
Library From a DOS Prompt**

**Building and Executing the
WebServ Demonstration System
From a DOS Prompt**

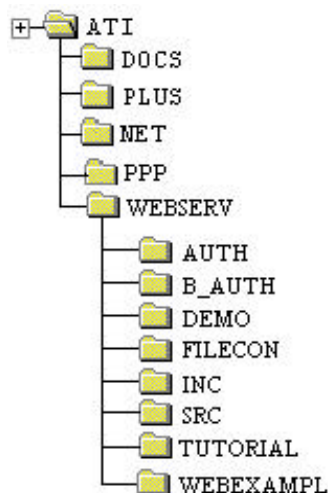
Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.



Introduction

This chapter will discuss the installation of the Nucleus WebServ source code, building the WebServ library, and building the demo. Nucleus PLUS, NET, a driver, and a file system must be installed before Nucleus WebServ is built. It is recommended that you verify these supporting libraries before attempting to build Nucleus Webserv. For more information concerning Nucleus PLUS refer to the chapters '*Nucleus PLUS*' and '*Nucleus PLUS Usage*' of this manual. For a discussion of the different drivers that can be used with Nucleus NET, see chapters '*Nucleus PQUICC Driver*' and '*Nucleus FEC860T Driver*' of this manual.

To install the Nucleus Webserv software, execute `SETUP.EXE` from the distribution software. All Nucleus products associated with this Motorola PowerPC 821/860 port using the Diab Data Tools must be installed in a common directory (`C:\ATI` by default). WebServ will require PLUS and NET to be present in this directory structure. Upon installation the demo application expects the PQUICC Ethernet driver and the Nucleus FILE system to also be in this directory structure. However, the demo can be modified so that other drivers may be used and the File Abstraction Layer found in `C:\ATI\FAL` can be modified so WebServ can use another file system. WebServ expects to be within this directory structure and will not build otherwise. After installation, the directory structure should resemble:



Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)

Building the Nucleus WebServ Library From a DOS Prompt

To build the library, execute the following command:

```
\ATI\WEBSERV> webserv <cr>
```

This will compile all the files of the WebServ library and put all of them into the library file, WEBSERV.A.

Building and Executing the WebServ Demonstration System From a DOS Prompt

Before building the WebServer demo (WEBDEMO.C) you will need to replace some information in the demo with information specific to your environment. These are the variables that will need to be modified:

```
#define printf          PRINTF
#define DEMO_IRQ        5
#define DEMO_IO_ADDR    0x0300L
#define DEMO_SERVER_IP_ADDR {100,200,300,400}
```

The symbol `printf` is a debugging function that is stubbed out by default. `DEMO_IRQ` and `DEMO_IO_ADDR` are not used by the PQUICC Driver, but are kept as an example for other drivers. Change `DEMO_SERVER_IP_ADDR` to the IP address of the embedded device.



By default, the demo can either use the File Abstraction Layer or the In Memory File system for the external WebPage support. All the demonstration directories within the Nucleus Webserv product are for use with the In-memory file system. Before building the demonstration application, if you want to use the In-memory file system please check the `\FAL\INC\IFILE.H` file and make sure that the `INCLUDE_FILE_LAYER` define is undefined. If you want to use Nucleus FILE make the `INCLUDE_FILE_LAYER` is defined in the `\FAL\INC\IFILE.H`. Once the `IFILE.H` driver has been changed, please rebuild it by going to the `\FAL` directory and running the `FAL.BAT` batch file.

If you are using the file abstraction layer, it expects Nucleus FILE. If you choose to use another file system, please refer to Chapter 4 of the *Nucleus Webserv Reference Manual*. The demo is also set up to use the PQUICC Driver, by default. To use another file system or network driver, see the documentation for that product.

IN_MEMORY_FILE_SYSTEM

Modify the `IFILE.H` in `\ATI\FAL\INC` to `#define INCLUDE_FILE_LAYER`, then rebuild the `FAL.A` library by executing:

```
\ATI\FAL> fal <cr>
```

Build the Webserv demo program by executing the `WEBDEMO.BAT` batch file. This will create the file `WEBDEMO.ELF` which can be downloaded to the debugger.

```
\ATI\WEBSERV\DEMO> webdemo <cr>
```

Once the demo has been started, execute the Web Browser of your choice (Internet Explorer or Netscape) and type in the IP address that you set your Network adapter to. For example, in `WEBDEMO.C` the IP address is set to `200.100.50.1`. The web browser should default to:

`http://your ip address/index.htm` page

Then follow the URL's to run through the different hyper links within the demo.

IF_NOT_USING_IN_MEMORY_FILE_SYSTEM (with Nucleus FILE)

Modify the `IFILE.H` in `\ATI\FAL\INC` to comment out `#define INCLUDE_FILE_LAYER`, then rebuild the `FAL.A` library by executing:

```
\ATI\FAL> fal <cr>
```

Build the Webserv demo program by executing the `WEBDEMOF.BAT` batch file. This will create the file `WEBDEMO.ELF` which can be downloaded to the debugger.

```
\ATI\WEBSERV\DEMO> webdemof <cr>
```

`WEBDEMO.ELF` is downloaded to the target, like the PLUS demo application.



To test Nucleus File support modify the `PS_CONF.H` file by commenting out the define `FS_IN_MEMORY`. Also comment out the `BASIC_AUTH` define (this will work here, but it is best not to make things more confusing). Now rebuild the WebServ library. Then rebuild the WebDemo executable. Then press *F5* to run the `WEBDEMO.EXE` application.

Now at the address line with the Web Browser type the following:

`http://your ip address/upload.htm.`

This will bring the `upload.htm` web page on the screen. Now go to the line filename to be saved on server and type `index.htm`. Then on the filename on computer click the browse button. Then go to the `.\WEBSERV\TUTORIAL\WEBFILES` directory and select `INDEX.HTM`. Next, click *Send File*.

You should get a response that the file, `INDEX.HTM` was saved on server. Now click back until you get to the *upload.htm* page and continue the above process for the following web pages, and images:

File to be named on server	Filename on computer(browse button)
A.GIF	.\WEBSERV\TUTORIAL\WEBFILES\A.GIF
SSI.SSI	.\WEBSERV\TUTORIAL\WEBFILES\SSI.SSI
GET.HTM	.\WEBSERV\TUTORIAL\WEBFILES\GET.HTM
POST.HTM	.\WEBSERV\TUTORIAL\WEBFILES\POST.HTM

You do not have to select the `UPLOAD.HTM` file for the fact it has already been placed in the root directory of the Ram Disk.

Now that you have all the files loaded into the ramdisk. Type in the URL for your IP ADDRESS\INDEX.HTM.

Run through all of the hyperlinks to verify that the function works correctly. This is the same pages as you did in the `TUTORIAL PS_FS.C` file, but running under Nucleus FILE.

This concludes the tests of the new functionality of the Nucleus WebServ product.

Select a `.GIF` for filename on computer, then click *submit*.



Miscellaneous Information

There is another demo that is contained in the `\TUTORIAL` directory. It also contains a `README.TXT` file that describes the files included in the demo.

In each of these directories there is a file called `PS_FS.C`. This file contains the in-memory file system structure for the embedded Web Server. This file is created with the windows based FileConvert Utility. This utility is provided in the `\FILECON` directory. In this directory, there contains three files. The first file is the `FILECONVERT.EXE` utility. The other two are the associated help files used with this utility.

The IP address of the server can be changed in the `WEBDEMO.C` file in the `Main_entry_task` function. If the IP address is changed then the internet address used to access the WebServer that is typed in is this address.



Nucleus CLIB

Introduction

**Building the Nucleus CLIB
Library From a DOS Prompt**

**Building and Executing the CLIB
Demonstration System From a
DOS Prompt**

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.



Introduction

The following chapter will discuss the installation of Nucleus CLIB, and building the Nucleus CLIB library.

Note: Nucleus PLUS must be installed, before Nucleus CLIB is installed. It is recommended that you build and execute the Nucleus PLUS demonstration application before attempting to build Nucleus CLIB.

To install the Nucleus CLIB software, execute `SETUP.EXE` from the distribution software. The Nucleus PLUS library and all other PLUS files must be in the PLUS directory as shown below. After installation, the directory structure should resemble:



CLIB depends on this directory structure and will not build correctly unless the batch files are modified.

Tools and Hardware

This release was tested with the following:

Tools	Version
Compiler:	DCC V4.2B
Assembler:	DAS V4.2B
Linker:	DLD V4.2B
Debugger:	Single Step On-Chip (MPC8XX) V7.4
Board:	Motorola MPC821/860 MBX, FADS and EST Boards*

(*programs for the EST board are compile-only.)



Building the Nucleus CLIB Library From a DOS Prompt

Build the CLIB library, `CLIB.LIB`, by executing the following command:

```
\ATI\CLIB> clib <cr>
```

Building and Executing the CLIB Demonstration System From a DOS Prompt

Build the Clib demo, `clibtest`, by executing the following command:

```
\ATI\CLIB\DEMO> demo <cr>
```

Download the demo in the same manner as you did the Nucleus PLUS demo. See the *Nucleus CLIB Reference* manual for more information on testing.





Nucleus EDE

Configuration for Nucleus
EDE

Demonstration System

Exclude From Build

Starting a Debugger Through
EDE

Setting the Tool Directory

Each chapter in this manual is specific to an individual Nucleus product. Each product is purchased separately, and may not have been received with your particular distribution.



Configuration of EDE

The PLUS EDE project was created with multiple configurations to enable the user to compile with each of the following three PowerPC 821/860 boards:

- FADS
- EST
- MBX

Select the appropriate configuration for your board by doing the following:

- Make sure the project is selected as the active project. This is done by right-clicking on the project and selecting '*Set as Active Project.*'
- Click on the '*Build*' menu, select '*Set Active Configuration.*'
- There are multiple configurations for each product. The name of the configuration should be self explanatory. Choose the appropriate one.
- Click *Ok*.

Demonstration System

The Nucleus products demos can be built in multiple configurations. Check the configurations for each demo you are attempting to build. The configurations should be self-explanatory.

Doing the following can change the project configurations:

- Make sure the desired project is selected as the active project. This is done by right-clicking on the project and selecting '*Set as Active Project.*'
- Click on the '*Build*' menu, select '*Set as Active Project.*'
- Choose the one of the configurations.
- Click *Ok*.



A simpler way of jumping between configurations is by adding the configuration window to your toolbar. This is done as follows:

- Click on the *'Tools'* menu, select *'Customize...'*
- Click the *'Toolbars'* tab
- Click on the *'Build'* checkbox, so that the *'Build'* toolbar is selected.
- Click *Ok*.
- You may also want to select dependencies. As an example, the *PlusPlus_Demo* depends on the *PLUS* library and *PLUSPLUS* library first being built.
- Therefore, you may set the dependencies by doing the following:
- Make sure the *'PlusPlus_Demo files'* project is selected as the active project.
- Click on the *'Project'* menu, select *'Dependencies...'*
- Click on the *PLUS* and *PLUSPLUS* checkboxes so that *PLUS* and *PLUSPLUS* are selected.
- Click *Ok*.

Setting the Tool Directory

In order to set your `<TOOLS_DIRECTORY>` macro, click on the *EDE Project Settings* toolbar shortcut. Click on the *Directories* tab. In the text window to the right of the *Tools Directory*, insure the path to your Diab Data directory is correct.

This path must include the drive, "diab", and then finally the diab version, for example: "c:\diab\4.3b".

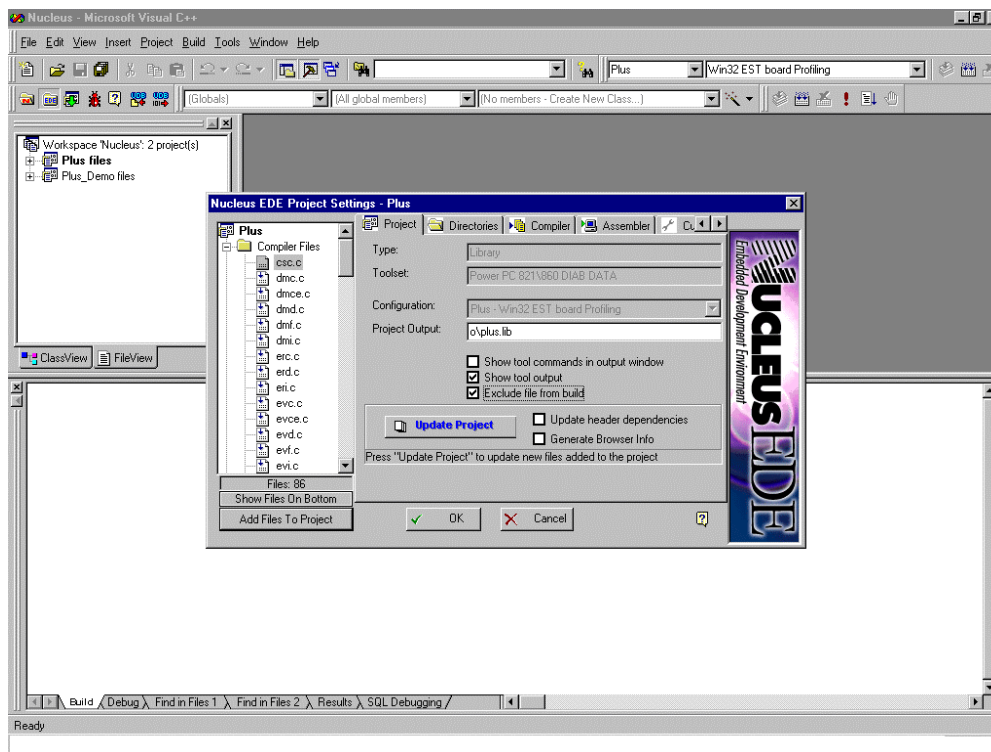


Exclude From Build

In some cases, a file may need to be excluded from the build. To accommodate this situation, Nucleus EDE provides a *"Exclude from build"* feature. This allows certain files of the project to be ignored during the build.

To set custom settings for a file:

Click on the EDE *"Project Settings"* toolbar shortcut. Click on the appropriate file in the *File/View* window. Click on the *"Project"* tab. Check the *"Exclude from build"* box.



Exclude from Build Screen



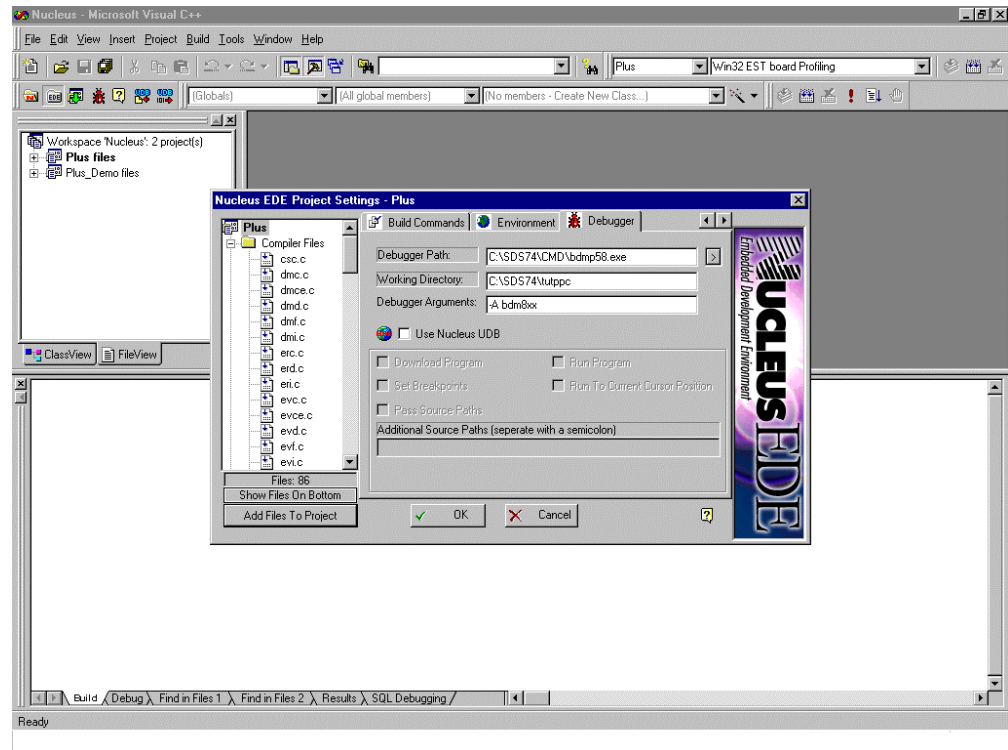
Starting a Debugger through EDE

Nucleus EDE provides a shortcut button that will automatically start your debugger. In the case that you change debuggers, these setting will need to be manually changed.

To change the debugger properties:

Click on the Nucleus EDE toolbar shortcut *"Project Settings."* Then, click on the *"Debugger"* tab. Enter the appropriate values for the *"Debugger Path," "Working Directory,"* and the *"Debugger Arguments."*

For example, to use the SDS Debugger:



Starting the SDS Debugger



